VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

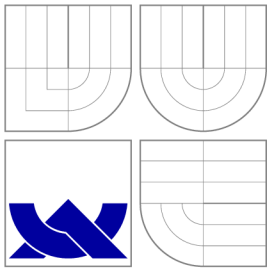
SLEDOVÁNÍ POHYBU SRDEČNÍHO SVALSTVA V ULTRAZVUKOVÉM ZÁZNAMU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

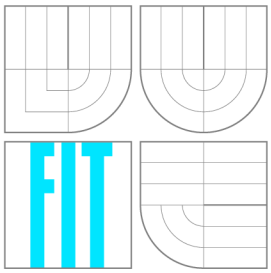
AUTOR PRÁCE
AUTHOR

Bc. JURAJ STRECHA

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SLEDOVÁNÍ POHYBU SRDEČNÍHO SVALSTVA V ULTRAZVUKOVÉM ZÁZNAMU

SPECKLE TRACKING ECHOCARDIOGRAPHY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JURAJ STRECHA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ŠTĚPÁN MRÁČEK

BRNO 2015

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2014/2015

Zadání diplomové práce

Řešitel: **Strecha Juraj, Bc.**
Obor: Bioinformatika a biocomputing
Téma: **Sledování pohybu srdečního svalstva v ultrazvukovém záznamu Speckle Tracking Echocardiography**
Kategorie: Počítačová grafika

Pokyny:

1. Nastudujte problematiku pořizování a zpracování ultrazvukových záznamů v medicíně. Zaměřte se na ultrazvuk srdce.
2. Navrhěte algoritmus, kterým lze sledovat pohyb dílčích částí srdečního svalstva v ultrazvukovém záznamu. Při návrhu algoritmu berte v úvahu celkový tvar srdce, například pomocí tzv. Active Shape Models.
3. Implementujte navržený algoritmus ve formě samostané aplikace. V grafickém výstupu aplikace by měla být patrná jak poloha dílčích sledovaných bodů tak celkový tvar srdce.
4. Otestujte aplikaci na databázi, která vám bude školitelem poskytnuta. Vyhodnoťte přesnost sledování pohybu a navrhněte možná vylepšení.

Literatura:

- H. Geyer, G. Caracciolo, H. Abe, S. Wilansky, S. Carerj, F. Gentile, H.-J. Nesser, B. Khandheria, J. Narula, and P. P. Sengupta, "Assessment of myocardial mechanics using speckle tracking echocardiography: fundamentals and clinical applications.," J. Am. Soc. Echocardiogr., vol. 23, no. 4, s. 351-369, Apr. 2010.
- S. Mondillo, M. Galderisi, and D. Mele, "Speckle-tracking echocardiography: a new technique for assessing myocardial function.," J. Ultrasound, vol. 30, s. 71-83, 2011.
- T. F. Cootes and C. J. Taylor, Statistical Models of Appearance for Computer Vision. Imaging Science and Biomedical Engineering, University of Manchester, 2004, p. 125.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Mráček Štěpán, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2014
Datum odevzdání: 27. května 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce se zabývá návrhem algoritmu a implementací programu, který v pořízeném ultrazvukovém videozáznamu srdce sleduje pohyb srdečního svalstva. Odhad polohy sledovaných bodů počítá metoda optického toku. K utvrzení správnosti polohy sledované struktury se používá statistický model Active Shape Model. Uživatel vyznačí strukturu srdečního oblouku a aplikace na dalších snímcích záznamu zobrazuje novou polohu bodů, které reprezentují nový deformovaný tvar.

Abstract

The thesis deals with proposal of an algorithm and implementation of a program that tracks a motion of the heart muscle in the captured ultrasound video of the heart. The point position estimation is calculated by optical flow method. The Active Shape Model method is used to confirm the accuracy of point's position tracking. The user annotates desired structure of the heart arch first and the application displays new points which represent a new deformed heart shape.

Klíčová slova

Echokardiografia, speckle tracking echocardiography, srdce, pravdepodobnostný model, modely aktivních tvarů, analýza hlavních komponent, optický tok

Keywords

Echocardiography, speckle tracking echocardiography, heart, probability model, active shape models, principal component analysis, optical flow

Citace

Juraj Strecha: Sledování pohybu srdečního svalstva
v ultrazvukovém záznamu, diplomová práce, Brno, FIT VUT v Brně, 2015

Sledování pohybu srdečního svalstva v ultrazvukovém záznamu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Štěpána Mráčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Juraj Strecha
26. května 2015

Poděkování

Děkuji vedoucímu mé práce panu Ing. Štěpánu Mráčkovi za odbornou pomoc při řešení problémů, které se vyskytly při řešení této práce a za čas, který mi věnoval. Chtěl bych také poděkovat panu MUDr. Ivanovi Sekovi za pomoc při výběru odborné studijní literatury v oblasti medicíny.

Obsah

Kapitola 1

Úvod

Kľúčovou úlohou medicíny je správne diagnostikovať všetky možné príčiny zhoršenia zdravotného stavu pacienta. Až na základe presnej identifikácie zdrojov problému môžu lekári nasadiť cielenú liečbu a ich príznaky zmierniť alebo úplne eliminovať. Rýchlo rozvýjajúcou sa a technicky zaujímavou oblasťou je kardiológia. Správna a včasná diagnóza môže zachrániť pacientovi život. Pri poruchách srdca človeka je najmenej invazívnou a najjednoduchšou snímkovacou metódou diagnostiky echokardiografia. Z vyhotovených snímkov, respektíve videozáznamu, je odborník schopný odhaliť odchýlky a poruchy štruktúry alebo cyklu srdca. V tomto prípade rádiológovi môže pomôcť strojové spracovanie signálu ultrazvukového prístroja pomocou automatizovaného vyznačenia a sledovania pohybu časti srdca, ktorú vyhodnotí ako potenciálne poškodenú.

Úlohou mojej diplomovej práce bolo navrhnúť a implementovať program, ktorý v ultrazvukovom zázname sleduje pohyb vyznačenej štruktúry srdca. V časti 2 zhrnieme základné poznatky o anatómii (2.1) a činnosti ľudského srdca (2.2), ktoré nám pomôžu interpretovať informácie nahraté pomocou echokardiografického prístroja. Čitateľa oboznámi aj s fyzikálnym princípom, na akom tento prístroj pracuje, a to v podkapitole 2.3. Kapitola 3 nám priblíži konkrétne informatické a matematické metódy spracovania a zobrazenia požadovanej štruktúry na natočenom videozázname z ultrazvuku. Sledovanie pohybu bodov dokáže metóda optického toku (Optical Flow) a čitateľovi je priblížená v podkapitole 3.3. V zašumenom prostredí záznamu nie je vždy možné detekovať všetky body správne, preto bolo nutné využiť ku kontrole odhadu pravdepodobnostný model tvaru ľavej komory srdca, ktorý počíta s tvarmi deformovateľnými v čase. Práca vychádza z článku Active Shape Models autorov Cootesa a Taylora, ktorí používajú techniky Procrustovej analýzy (podkapitola 3.4), analýzy hlavných komponent (Principle Component Analysis) (podkapitola 3.5) a modely aktívnych tvarov (Active Shape Models) (podkapitola 3.6) pre vyhodnotenie polohy bodov v postupnosti snímkov. Na základe analýzy projektu a zadaných požiadaviek som mal vytvoriť návrh riešenia. Popisuje ho kapitola 4, kde sa dočítame aj o navrhnutom algoritme. Implementačné detaily môjho riešenia nájde čitateľ v časti 5 a otestovanie funkčnosti algoritmu a programu na poskytnutých vzorových dátach v 6. V kapitole 6 tiež nájdeme zhodnotenie dosiahnutých výsledkov práce. V závere naznačíme aj možné vylepšenia, ktorými by sa riešenie mohlo ešte spresniť.

Kapitola 2

Anatomický rozbor ľudského srdca

Využitie najnovšej výpočtovej techniky zohráva významnú úlohu v oblasti modernej medicíny a diagnostiky. Na každom oddelení v medicínskych zariadeniach už dnes nájdeme prístroj, ktorý využíva počítačovú techniku a prácu s informáciami. Dáta získavame klasickou cestou postavenou na princípoch fyziky a chémie (röntgenové žiarenie, ultrazvuk, reakcia substrátu na protilátky, premena látok, pozorovanie mikroskopiou). Takto namerané informácie je možné digitalizovať a uložiť do databázy, či priamo digitálne zaznamenať pokročilými technológiami analógovo – digitálneho prevodu. Množstvo takto získaných poznatkov v dnešnej dobe rastie neuveriteľnou rýchlosťou vďaka cenovej dostupnosti a popularite výpočtovej techniky v každodennom živote. Aby sme boli schopní s dátami efektívne pracovať tak, aby nám uľahčovali prácu a pomáhali pri riešení problémov, musíme ich byť schopní v krátkom čase a efektívne spracovať. V ďalšom texte nájdeme popis toho, čo vidíme na echokardiografickom zázname, ako aj spôsob, akým sa záznam vytvára.

2.1 Základná anatómia srdca

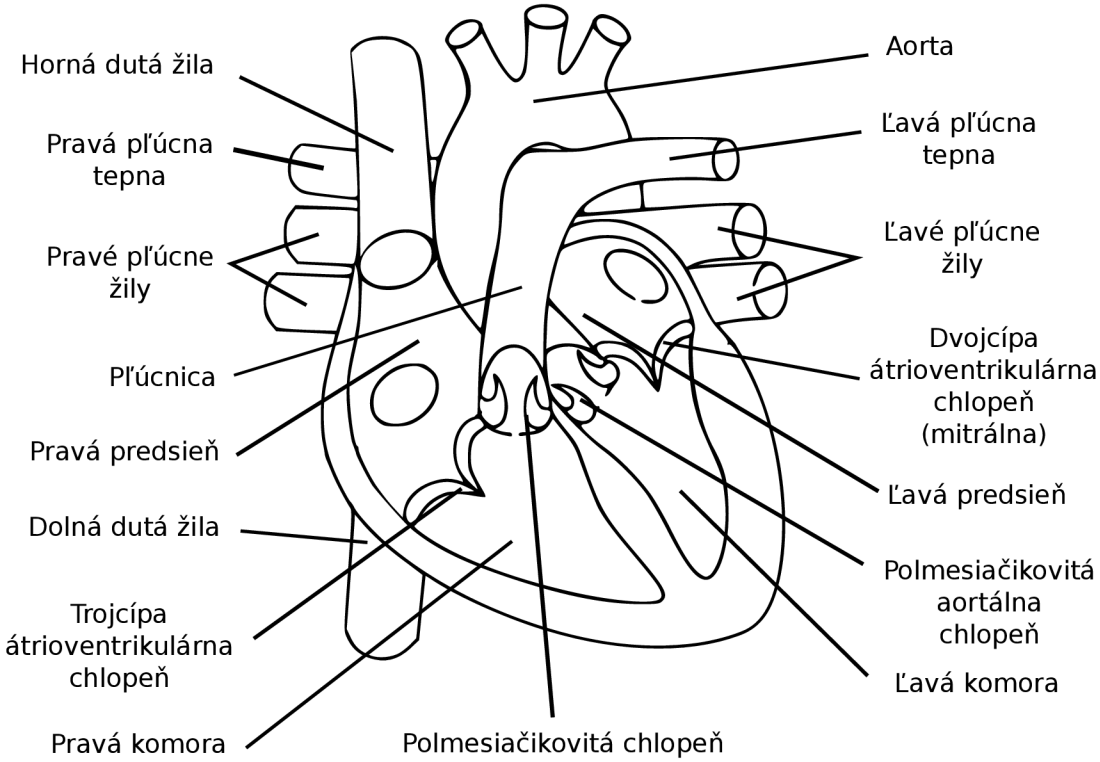
Ľudské srdce ako nepárový orgán funguje v rámci srdcovocievnej (kardiovaskulárnej) sústavy. Jej úlohou je predovšetkým zabezpečiť transport kyslíka a výživných látok do orgánov a tkanív, a odvádzať z nich oxid uhličitý a odpadové produkty látkovej premeny. Ďalej distribuuje v organizme hormóny, ochranné látky a reguluje telesnú teplotu. Celú túto sústavu tvorí krv, sústava krvných ciev a srdce ako orgán udržiavajúci krv v obehu. Srdce je uložené v hrudnom koši a natočené tak, že 2/3 objemu sú vľavo a 1/3 vpravo od stredovej roviny [16]. Je obalené väzivovým vakom zvaným osrdcovník (pericardium). Veľkosť srdca jedincov variuje a často závisí od veku, pohlavia a funkčného zaťaženia. Zodpovedá zhruba veľkosti päste človeka, ktorému patrí [16].

Na biologickej úrovni je srdce tvorené špeciálnym druhom priečne pruhovaného svalstva – myocardiom. Ďalej rozlišujeme tri typy srdcového svalstva – predsieňovú svalovinu, komorovú svalovinu a špeciálnu excitačnú svalovinu. Predsieňový a komorový typ označujeme aj ako pracovný myokard, pretože sa kontrahujú podobne ako kostrové svaly a prostredníctvom kontrakcie vykonávajú prácu [12]. Prácou v tomto kontexte rozumieme udržiavanie obehu krvi v systéme.

Srdce pozostáva zo štyroch dutín, z toho dve sú predsiene: pravá (atrium dextrum) a ľavá predsieň (atrium sinistrum) a dve komory: pravá (ventriculum dexter) a ľavá (ventriculum sinister) [16]. Pravú a ľavú časť srdca oddeľuje prepážka Septum. Spomenuté štruktúry sú dobre pozorovateľné práve na echokardiografickom zázname. Významným objektom vyše-

trovania sú aj chlopne. Pracujú na princípe mechanizmu, ktorý dovoľuje prietok krvi len jedným smerom. Delíme ich do dvoch skupín – cípové (valvae atrioventriculares) a polmesiačikové chlopne (valvae semiluminares) [16]. Átrioventrikulárne sa ďalej delia podľa počtu cípov, z ktorých sú zložené, na dvojcípe a trojcípe.

Do dutiny pravej predsiene vyúsťujú horná a dolná dutá žila. Privádzajú neokysličený krv z celého tela. Predsieň je od pravej komory oddelená trojcípou átrioventrikulárnou chlopňou. Z pravej komory smeruje vývod do tepny pľúcnice. Uzatvára sa v diastolickej fáze srdcového cyklu polmesiačikovitou chlopňou. Ľavá predsieň je približne štvorhranného tvaru, čo je dané štyrmi pľúcnymi otvormi [16], ktoré nie sú uzavreté žiadnymi chlopňami. Pľúcne žily cez tieto miesta dodávajú do srdca okysličenú krv z pľúc. Od ľavej komory oddeľuje predsieň dvojcípa átrioventrikulárna chlopňa – označovaná tiež mitrálna chlopňa (valva mitralis). K ľavej komore je pripojená tepna srdcovnica, do ktorej jednosmerne cez semiluminárnu chlopňu valva aortae prúdi okysličená krv. Funkčný popis všetkých spomenutých štruktúr nájdeme popísaný v podkapitole 2.2. Schématický nákres srdca je znázornený na Obrázku 2.1.



Zdroj:http://upload.wikimedia.org/wikipedia/commons/a/a7/Heart.svg

Obrázek 2.1: Štruktúra srdca.

Špeciálna časť pravej predsiene sa nazýva sínusový uzol. Jeho vlákna sa vyznačujú najväčšou samoexcitáciou, a preto udávajú rytmus (pacemaker) pre celý myokard [4]. Riadia tým srdcový cyklus popísaný v ďalšej časti textu.

Srdce charakterizujú nasledujúce fyziologické vlastnosti [4]:

- **Vzrušivosť (excitabilita)** – schopnosť srdca reagovať na podnety kontrakciou, schopnosť myocytov (buniek svalového tkaniva) excitovať sa aspoň prahovým podnetom
- **Autorytmicita** – vzruchy v špecializovaných bunkách sínusového uzla srdca vznikajú spontánne a v pravidelných intervaloch
- **Vodivosť (konduktibilita)** – vzruchy vytvorené v sínusovom uzle sa šíria na ostatné štruktúry srdca špeciálnou vodivou svalovinou
- **Sťažlivosť (kontraktilita)** – schopnosť srdcového svalu kontrahovať sa

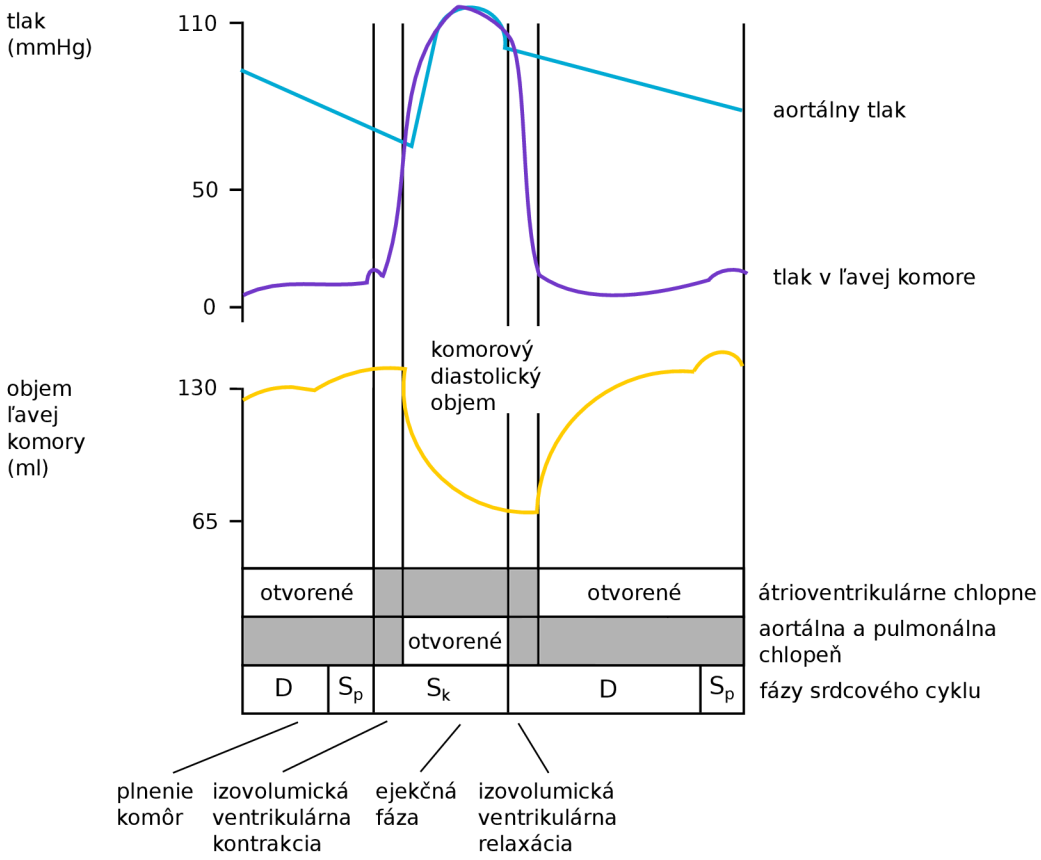
2.2 Srdcový cyklus

Srdce funguje ako pumpa. Jeho činnosť je neustále sa opakujúci cyklický dej. Jeden srdcový cyklus sa skladá z relaxácie, nazývanej diastola a kontrakcie, ktorej hovoríme systola [4]. Počas diastoly sa srdce naplní krvou [12]. Excitácia – vzrušenie srdcového svalu vedie rýchlo k jeho mechanickej kontrakcii – systole [4]. Počas nej je okysličená krv vytláčaná zo srdca do celého tela. Systola a diastola sa pravidelne striedajú, čím prečerpávajú krv medzi malým krvným obehom (medzi srdcom a pľúcami) a veľkým (telovým) krvným obehom. Dôležité je, aby srdcové chlopne fungovali spoľahlivo a zabránili krvi v návrate naspäť do predsiení, respektíve komôr.

Srdcový cyklus pozostáva z nasledujúcich fáz [4]:

- **Plniaca fáza** – Trvá 2/3 času predsieňovej diastoly. Počas komorovej systoly sa v predsieňach hromadí pritekajúca krv zo žíl. Po skončení systoly a dosiahnutí mierne vyššieho tlaku v predsieňach sa otvoria átrioventrikulárne chlopne a krv začne rýchlo vtekať do komôr – jedná sa o fázu rýchleho plnenia. Aj po vyčerpaní veľkého množstva krvi z predsiení zostávajú chlopne medzi predsieňami a komorami naďalej otvorené a pritekajúca krv prúdi rovno do komôr – fáza pomalého plnenia.
- **Systola predsiení** – V poslednej tretine diastoly sa predsiene kontrahujú. Dokončí sa tým plnenie komôr, dosiahli takzvaný enddiastolický objem krvi.
- **Izovolumická kontrakcia** – Fáza, kedy sa svalovina komory kontrahuje, ale ešte nedochádza k vyprázdneniu. So zvýšením tlaku v komorách sa uzavrú átrioventrikulárne chlopne.
- **Ejekčná fáza** – Tlak v ľavej komore presiahne hodnotu diastolického tlaku v aorte (asi 10,7 kPa, 80 mmHg) a pľúcnej tepne (asi 10% diastolického tlaku v aorte [12]) následkom čoho sa otvoria semilunárne chlopne a z komôr vytečie asi 70% krvi. Táto časť trvá približne 1/3 fázy ejekcie a nazývame ju fáza rýchlej ejekcie. Zvyšné 2/3 času, pomenovaná pomalá ejekčná fáza, odtečie zvyšných 30% objemu krvi v komore.
- **Izovolumická relaxácia** – Dochádza k uvoľneniu vnútrokomorového tlaku. Tlak v artériách je vyšší ako v komorách a uzavrú sa aortálna a pulmonálna chlopňa. Po klesnutí tlaku na diastolické hodnoty sa otvoria átrioventrikulárne chlopne a začne ďalší cyklus.

Na Obrázku 2.2 sú znázornené všetky spomenuté fázy. Vidíme stav objemu ľavej komory meniaci sa s postupom fáz a tiež grafické znázornenie vývoja tlaku v komore a aorte. Nákres ilustruje aj otváranie a zatváranie chlopní v jednotlivých častiach cyklu.



Obrázek 2.2: Udalosti a tlak ľavej komory a aorty počas jedného cyklu.

2.3 Echokardiografia

Pojem echokardiografia označuje ultrazvukové vyšetrenie srdca. Pre snímkovanie pomocou ultrazvuku sa v praxi často používa aj synonymné označenie sonografia. Jedná sa o neinvazívnu metódu diagnostického vyšetrenia, ktorá sa používa aj pre zobrazenie iných vnútorných orgánov ako len srdca, napríklad obličiek. Na rozdiel od röntgenového vyšetrenia nie je pacient vystavený rádioaktívnym vlnám, ktoré trvalo poškodzujú tkanivá. Jedinou diskomfortnou časťou zákroku môže byť prvotné priloženie sondy natretej kontaktným gélom na telo vyšetrovaného subjektu. Vyškolený personál medicínskeho zariadenia – rádiológ vidí v reálnom čase stav tkanív pacienta na monitore prístroja. Echoprístroje súčasnosti sú prenosné, nepotrebujú byť v špeciálnej miestnosti medicínskeho zariadenia, a tak pacient nemusí kvôli vyšetreniu opustiť svoju izbu ani lôžko.

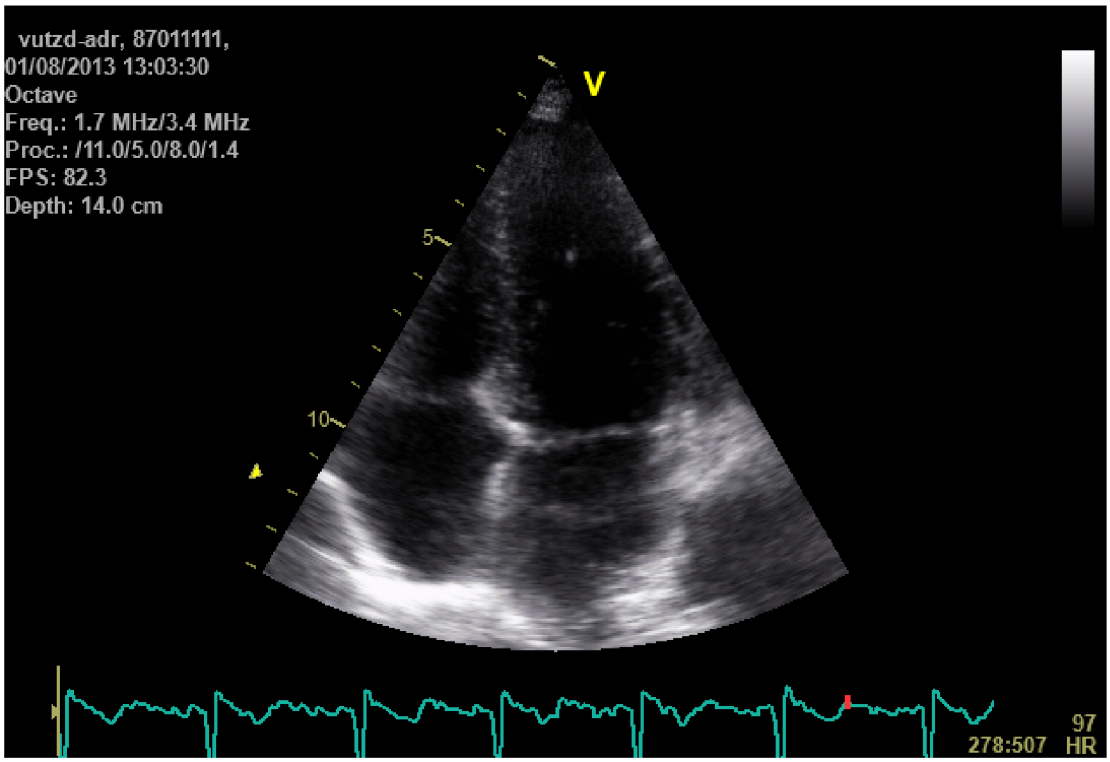
Ultrazvuková diagnostika používa zvukové vlny rovnaké ako počuteľný zvuk [20], ale na vyššej frekvencii. Človek dokáže rozpoznať zvuky vo frekvenčnom spektre od 20 Hz do 20 kHz (novorodenci o niečo viac ako 20 kHz, vekom sa horný limit znižuje na 15-17 kHz u dospelého človeka) [18]. Diagnostické ultrazvukové vlny sa pohybujú v intervale 2 MHz až 15 MHz [17] a to hlavne z dôvodu, že sa najlepšie šíria v kvapalinách, zatiaľ čo v pevných látkach a plynoch sú výrazne tlmené. Rozličné sondy vysielajú na rôznych frekvenciách.

K vyšetreniu hlbších štruktúr sa používajú frekvencie 2-5 MHz, pre zobrazenie oblastí bližšie k povrchu tela 5-15 MHz. Zvukové vlny na nižšej frekvencii sa dostanú aj za roh, pričom vyššie frekvencie letia viac v priamom smere a, čo je dôležité pre ultrasonografiu, dokážu sa odraziť od malých objektov. To súvisí s ich krátkou vlnovou dĺžkou oproti vlnám na nižších frekvenciách. Vlnová dĺžka λ je nepriamo úmerná frekvencii f v závislosti na rýchlosti zvuku c v prostredí podľa vzťahu 2.1 [20].

$$\lambda = c/f \tag{2.1}$$

Zdrojom zvuku je piezoelektrický kryštál uložený v sonde, ktorý pôsobením striedavého prúdu deformuje svoj tvar, čím vysiela (asi 0,5% času) ultrazvukové vlny. Opačný princíp sa využíva k zachyteniu odrazu (zvyšných 99,5% času) [17]. Hranice tekutého prostredia (kam môžeme pre vysoký obsah vody zaradiť aj mäkké tkanivá) s kosťou alebo plynom predstavujú tak výrazné rozhranie, že na ňom dochádza k odrazu takmer všetkého ultrazvukového vlnenia. Preto prakticky nie je možné vyšetrovať orgány uložené za skeletom alebo plynom. To je tiež dôvod, prečo je nutné používať kontaktné gély na kožu – vďaka nim je odstránená tenká vrstva vzduchu medzi pokožkou a sondou, ktorá by bránila prechodu vĺn do vyšetrovanej oblasti [17].

Najčastejšie používaným typom ultrazvukového záznamu je dynamický B-mód (brightness mode). Pri ňom obraz vzniká zachytením veľkého množstva vedľa seba umiestnených odrazov, ktorým je v závislosti na intenzite odrazu priradený na monitore príslušný odtieň šedej farby [17]. Príklad obrazu získaného ultrazvukovým prístrojom v B-móde ukazuje Obrázok 2.3. Nevýhodou tohto druhu zobrazovania je pomerne veľká úroveň šumu spôsobená tým, že zvuková vlna sa neodráža vždy v smere, z ktorého bola vyslaná, ale na zakrivenom povrchu dochádza k odrazom do strán, preč od sondy.



Obrázek 2.3: Pohľad na ľudské srdce pomocou ultrazvuku v B-móde.

Kapitola 3

Využité technológie

Kapitola približuje matematické princípy, metódy a modely, ktoré sme pri riešení zadania použili. Analytické vyhodnotenie parametrov transformácie dvoch tvarov tak, aby sa čo najlepšie zarovnali na seba pomocou Singular Value Decomposition (SVD). Rovnomerné rozloženie bodov tvaru a hladké vykreslenie zabezpečí parametrická krivka Catmull-Rom splajn. Optický tok (Optical Flow) ako samostatnú časť pre odhad pohybu a Procrustovu analýzu s analýzou hlavných komponent (Principle Component Analysis) ako súčasť pravdepodobnostného modelu aktívnych tvarov (Active Shape Models).

3.1 Singular Value Decomposition

Rozloženie matice na časti, ktoré nie sú zrejmé na prvý pohľad, je častým krokom vo veľkom množstve algoritmov. Metódu môžeme použiť na reálne aj komplexné matice, častejšie sa však pracuje s reálnymi. Ideou je fakt, že obrazom transformácie jednotkovej gule S (angl. *unit sphere*) pomocou matice A s rozmermi $m \times n$ vznikne hyperelipsa [22] AS. Uvažujme hyperelipsu $\mathcal{R}^m$ ako zobecnenie 2D epipsy v m-dimenzionálnom priestore. Získame ju tak, že povrch jednotkovej gule m-rozmerného priestoru roztiahneme pomocou činiteľov $\sigma_1, \ldots, \sigma_m$ v ortogonálnych smeroch $u_1, \ldots, u_m$ [22]. Považujme u_i za jednotkové vektory. Potom vektory $\sigma_i u_i$ s dĺžkami $\sigma_1, \ldots, \sigma_m$ sú hlavné poloosy roztiahnutej hyperelipsy. Pre dvojrozmerný priestor predstavuje jednotkovú guľu kružnica so stredom v počiatku kartézskej súradnicovej sústavy a polomerom 1. Deformáciou vznikne natočená 2D elipsa roztiahnutá v hlavných poloosách. Pre maticu $m \times n$ hodnosti n bude nenulových práve n činiteľov σ_i.

Pre maticu A definujme ľavé singulárne vektory $U = u_1, \ldots, u_n$ ako hlavné poloosy AS, pravé singulárne vektory $V = v_1, \ldots, v_n$ ako jednotkové vektory jednotkovej gule S a $\Sigma = \sigma_1, \ldots, \sigma_n$ ako dĺžky hlavných poloosí AS. Hodnoty σ_i sú zoradené zostupne a k nim sú zodpovedajúco zoradené vektory u_i. Hlavný vzťah medzi pravými a ľavými singulárnymi vektormi zapíšeme ako

$$Av_j = \sigma_j u_j \text{ [22]}. \tag{3.1}$$

Maticový zápis výrazu je $AV = U\Sigma$. Σ je $n \times n$ diagonálna matica, U je $m \times n$ matica, V je $n \times n$, obe s ortonormálnymi stĺpcami. Keďže je V ortogonálna matica, vynásobením V s V^T dostaneme jednotkovú maticu s hodnotami 1 na diagonále a nulami inde. Vynásobením vzťahu V^T prevedieme vzťah na výsledný rozklad

$$A = U\Sigma V^T. \tag{3.2}$$

3.2 Catmull-Rom splajn

Skupina kriviek Catmull-Rom sa radí medzi kubické interpolačné splajny. Prekladajú zadané kontrolné body hladkou krivkou definovanou polynómom tretieho rádu v tvare

$$y = a + bx + cx^2 + dx^3. \tag{3.3}$$

Koeficienty a, b, c, d obecnej Fergusonovej parametrickej kubickej krivky

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.4}$$

zistíme podľa [14] vyriešením štyroch lineárnych rovníc

$$\begin{aligned} P(0) &= a_0 \\ P(1) &= a_0 + a_1 + a_2 + a_3 \\ P'(0) &= a_1 \\ P'(1) &= a_1 + 2a_2 + 3a_3, \end{aligned}$$

kde $P(0), P(1)$ sú riadiace body a $P'(0), P'(1)$ príslušné sklony. Dosadením vypočítaných koeficientov do 3.4 vznikne vzťah pre výpočet Fergusonovej krivky

$$P(t) = (1 - 3t^2 + 2t^3)P(0) + (3t^2)P(1) + (t - 2t^2 + t^3)P'(0) + (-t^2 + t^3)P'(1). \tag{3.5}$$

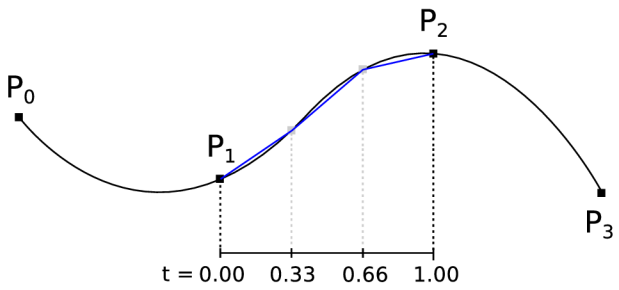
Catmull-Rom medzi dvomi bodmi P_i a P_{i+1} priraďuje kontrolným bodom dotyčnice s krivkou ako

$$\frac{P_{i+1} - P_{i-1}}{2}, \frac{P_{i+2} - P_i}{2} \text{ [14]}. \tag{3.6}$$

Dosadením vzťahu pre dotyčnice do rovnice výpočtu bodu Fergusonovej krivky dostaneme po úprave výsledný vzťah pre bod Catmull-Rom splajnu s parametrom t

$$P(t) = \frac{1}{2} * ((-t^3 + 2t^2 - t)P(0) + (3t^3 - 5t^2 + 2)P(1) + (-3t^3 + 4t^2 + t)P'(0) + (t^3 - t^2)P'(1)). \tag{3.7}$$

Typy kriviek v skupine sa líšia rozložením parametrov. Základná varianta Catmull-Rom používa uniformné rozloženie. Parameter $t \in < 0; 1 >$ má rozložené hodnoty uniformne v danom intervale. Od ich počtu závisí, koľko bodov krivky sa vypočíta medzi dvoma kontrolnými bodmi. Zjednodušene povedané, aká hladká bude krivka. Rozdelenie na tri intervaly vidíme na Obrázku 3.1, kde môžeme pozorovať viditeľné rovné úsečky medzi vypočítanými bodmi.



Obrázek 3.1: Catmull-Rom interpolačná krivka (modrá).

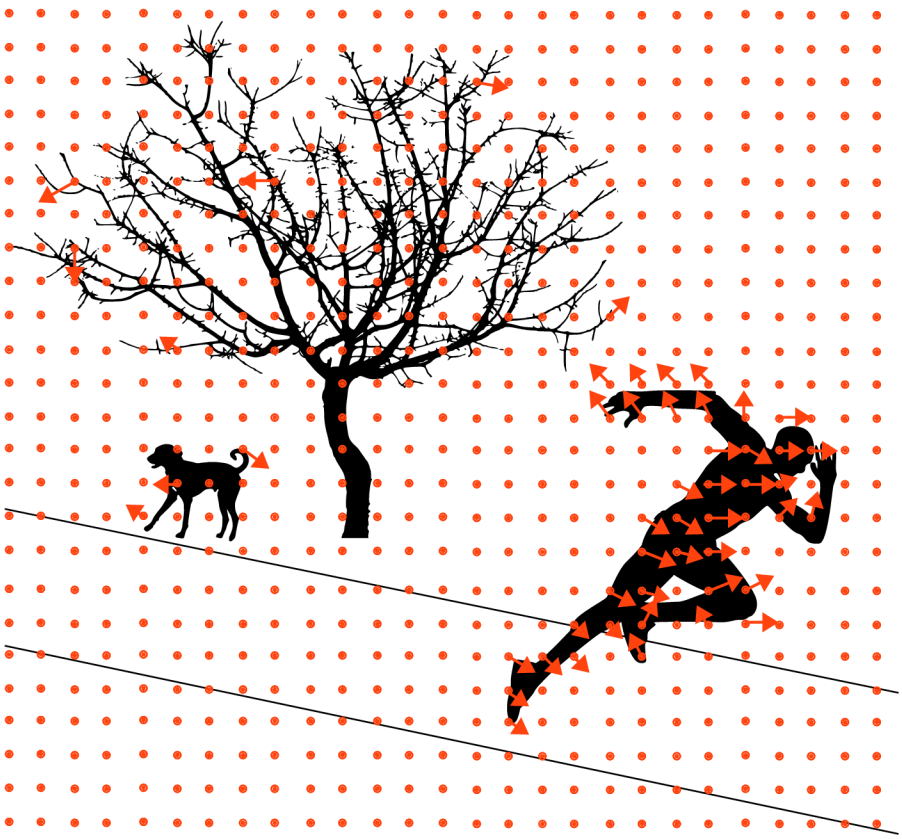
Kontrolné body ovplyvňujú iba relatívne malé okolie a jednotlivé časti majú explicitne definované polynómy pre výpočet bodov [25].

Catmull-Rom skladá výslednú krivku zo štyroch bodov tak, že medzi dvomi nasledujúcimi riadiacimi bodmi p_i a p_{i+1} vytvorí krivku ovplyvnenú bodmi p_{i-1} a p_{i+2} [10] tak, ako vidíme na Obrázku 3.1 Je dobré rozšíriť množinu kontrolných bodov o počiatočný a koncový bod ich skopírovaním na začiatok a koniec zoznamu. Skladaním po častiach dostávame získame výsledný tvar.

Pre jednoduchosť výpočtu, hladký priebeh a interpolačnú vlastnosť sú rozšírené v oblasti počítačovej grafiky, modelovania a animácií. Nevýhodou je, že v základnej implementácii za určitých podmienok vytvárajú sľučky a pretínajú sa.

3.3 Optical Flow

Optický tok (optical flow) je vzor zdanlivého pohybu objektu, povrchu alebo okraja vo vizuálnej scéne spôsobený relatívnym pohybom medzi pozorovateľom a scénou [23]. Tok v obraze sa určuje odhadom. Väčšina prístupov pre odhad optického toku je založená na zmenách jasu medzi dvoma po sebe nasledujúcimi scénami sekvencie [1]. Na trojrozmernej scéne sa objekt alebo kamera pohybuje po trojrozmernej trase $\vec{X}(t)$. Po premietnutí do obrazovej roviny vytvárajú body dvojrozmernú trasu $\vec{x} \equiv (x(t), y(t))^\intercal$, okamžitý smer s rýchlosťou $d\vec{x}(t)/dt$ [6]. Tieto 2D rýchlosti každého viditeľného bodu tvoria 2D pohybové pole [9]. Príklad poľa optického toku vidíme na Obrázku 3.2.



Obrázek 3.2: Pohybové pole optického toku.

Bolo navrhnutých hneď niekoľko prístupov pre odhad pohybu, ako korelácia alebo block – matching, feature tracking a metódy založené na energii [6]. Bližšie si priblížime metódy založené na gradiente, pretože jedna z nich je použitá v implementácii knižnice *OpenCV*.

Uvažujme, že intenzity každého bodu sú prenesené medzi dvoma po sebe nasledujúcimi obrázkami videosekvencie, čo môžeme vyjadriť ako

$$I(\vec{x}, t) = I(\vec{x} + \vec{u}, t + 1), \tag{3.8}$$

kde $I(\vec{x}, t)$ je intenzita obrázku ako funkcia bodu $\vec{x} = (x, y)^\intercal$ so súradnicami x, y v čase t a $\vec{u} = (u_1, u_2)^\intercal$ je rýchlosť. Vzťah označujeme ako nemennosť jasu (brightness constancy). V praxi však rovnosť platí iba zriedkavo a jas jednotlivých bodov môže ovplyvniť zmena nasvietenia scény, tiene, reflektovanie povrchu a významne tiež šum v obrazovom signále. Pre odvodenie mechanizmu odhadu 2D rýchlosti vyjadríme posun obrazu ako aproximáciu Taylorovým rozvojom prvého rádu

$$I(\vec{x} + \vec{u}, t + 1) \approx I(\vec{x}, t) + \vec{u}. \bigtriangledown I(\vec{x}, t) + I_t(\vec{x}, t), \tag{3.9}$$

kde $\bigtriangledown I \equiv (I_x, I_y)$ a I_t značí priestorovú a časovú parciálnu deriváciu obrázku, $\vec{u} = (u_1, u_2)^\intercal$ predstavuje 2D rýchlosť [6]. Keď zanedbáme členy vyššieho rádu Taylorovho rozvoja a dosadíme aproximáciu do (3.8), získame

$$\bigtriangledown I(\vec{x}, t).\vec{u} + I_t(\vec{x}, t) = 0. \tag{3.10}$$

Rozvinutím $\bigtriangledown, \vec{u}$ a presunutím jedného člena rovnosti na pravú stranu výrazu dostaneme prehľadnejší vzťah

$$I_x(\vec{x}, t).u_1 + I_y(\vec{x}, t).u_2 = -I_t(\vec{x}, t). \tag{3.11}$$

Vzťah (3.10) sa v literatúre nazýva gradient constraint equation [6]. Keďže $\vec{u}$ je vektor s dvoma zložkami – neznámymi u_x a u_y, nie je možné vypočítať ich z jednej rovnice (3.10), v ktorej obe neznáme figurujú. Spomenutý jav bol pomenovaný ako problém clony (aperture problem) [24]. Existuje viacero prístupov, ktoré túto nejednoznačnú situáciu riešia. V praxi sa najčastejšie používa diferenčná metóda, ktorú navrhli Bruce D. Lucas a Takeo Kanade [15] a v ďalšom texte si ju viac priblížime. Verziu ich postupu implementuje aj knižnica pre prácu s obrazom *OpenCV*.

Predpokladajú, že pole optického toku v malom susedstve istého obrazového bodu x je konštantné. Používajú pri tom okolie – okno $n \times n$, v ktorom uvažujú všetky pixely, čím vznikne viac rovníc ako neznámych.

$$\begin{aligned}
I_x(\vec{x_1}, t).u_1 + I_y(\vec{x_1}, t).u_2 &= -I_t(\vec{x_1}, t) \\
I_x(\vec{x_2}, t).u_1 + I_y(\vec{x_2}, t).u_2 &= -I_t(\vec{x_2}, t) \\
&\vdots \\
I_x(\vec{x_n}, t).u_1 + I_y(\vec{x_n}, t).u_2 &= -I_t(\vec{x_n}, t)
\end{aligned}$$

Pri odhade môžeme hovoriť o optimalizačnej úlohe, ktorej riešenie by malo spĺňať určitú podmienku. Jednou z nich je rovnica (3.11), ďalšie sú novovzniknuté rovnice pre okolie. S viacerými podmienkami nemusí existovať také $\vec{u}$, ktoré by ich spĺňalo všetky naraz. Namiesto toho sa snažíme nájsť $\vec{u}$ riešenie, ktoré minimalizuje chyby podmienok [6]. Využijeme známu metódu najmenších štvorcov a hľadáme najmenšie

$$E(\vec{u}) = \sum_{\vec{x}} g(\vec{x})[I_x(\vec{x_1}, t).u_1 + I_y(\vec{x_1}, t).u_2 + I_t(\vec{x_1}, t)]^2, \tag{3.12}$$

kde $q(\vec{x})$ je váhová funkcia bodov okolia [6]. Často sa používa normálne (Gaussovo) rozloženie, kde stred okolia viac ovplyvňuje výsledok.

Minimálne $E(\vec{u})$ nájdeme v bodoch, kde parciálne derivácie podľa $\vec{u}$ sú rovné nule [6]

$$\begin{aligned}\frac{\partial E(u_1,u_2)}{\partial u_1} &= \sum_{\vec{x}} g(\vec{x})[u_1 {I_x}^2 + u_2 I_x I_y + I_x I_t] = 0\\ \frac{\partial E(u_1,u_2)}{\partial u_2} &= \sum_{\vec{x}} g(\vec{x})[u_2 {I_y}^2 + u_1 I_x I_y + I_y I_t] = 0,\end{aligned}$$

čo môžeme prepísať do maticového tvaru

$$M\vec{u} = \vec{b}, \tag{3.13}$$

kde

$$M = \begin{bmatrix} \sum g{I_x}^2 & \sum g I_x I_y \\ \sum g I_x I_y & \sum g {I_y}^2 \end{bmatrix}, \vec{b} = -\begin{pmatrix} \sum g I_x I_t \\ \sum g I_y I_t \end{pmatrix}. \tag{3.14}$$

Ak je hodnosť matice rovná dvom, potom odhad pomocou najmenších štvorcov vypočítame ako $\hat{u} = M^{-1}\vec{b}$.

Aby sme vyriešili problém uzávierky (aperture problem), nový bod vždy hľadáme v definovanom okolí (integračnom okne). Má tvar štvorca s typickými dĺžkami strán 5, 7, 9, 11, 13 a 15 [2]. Ďalej hovoríme o takzvanej pyramídovej implementácii metódy Lucas-Kanade. Táto verzia je implementovaná ako funkcia knižnice *OpenCV*. Rieši pohyby pixelov medzi dvoma obrázkami väčšie, ako je zadané integračné okno. Obraz I definujeme na viacerých úrovniach. Úroveň 0 značí, že obraz má najvyššie rozlíšenie (pôvodné) $n_x^0 = n_x$ a $n_y^0 = n_y$. Ďalšie úrovne pyramídovej reprezentácie sa budujú rekurzívne, I^1 získame z I^0, I^2 z I^1 a tak ďalej. Označme $L = 1, 2, \ldots$ úroveň pyramídy a I^{L-1} obraz na úrovni $L-1$. Výšku a šírku obrazu predstavujú n_x^{L-1} a n_y^{L-1}. Potom je I^{L-1} podľa [2] definované ako

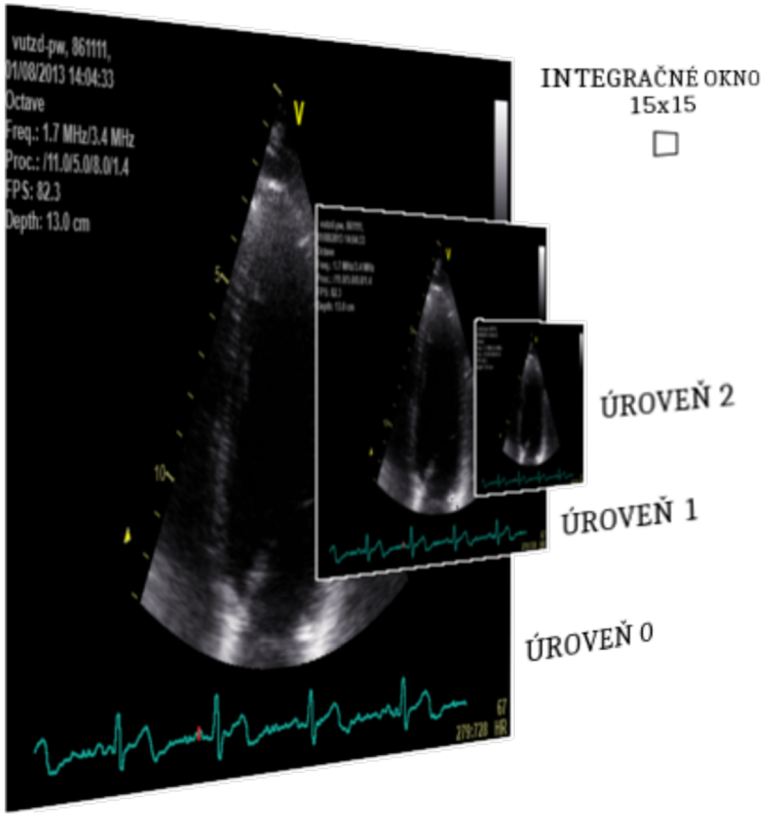
$$\begin{aligned}I^L(x,y) = \frac{1}{4}I^{L-1}(2x,2y) + \\ \frac{1}{8}(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)) + \\ \frac{1}{16}(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y-1)I^{L-1}(2x+1,2y+1)).\end{aligned} \tag{3.15}$$

Rovnica 3.15 je definovaná iba pre hodnoty $0 \leq 2x \leq n_x^{L-1}$ a $0 \leq 2y \leq n_y^{L-1}$. V praxi sa najčastejšie využívajú výšky pyramídy L_m 2, 3, 4 [2]. Volíme výšku podľa predpokladaného maximálneho optického toku v obraze.

Postup pri odhade optického toku pyramídovej implementácie je nasledujúci: výpočet optického toku pre level L_m, najmenšie rozlíšenie obrazu, propagácia výsledku vo forme počiatočného odhadu posunu pixelov o jednu úroveň nižšie, zjemnenie optického toku. Postup opakujeme pokiaľ nedosiahneme úroveň L_0.

3.4 Procrustova analýza

Analýza štatisticky vyhodnocuje sadu tvarov. Dokáže zarovnať prvky množiny vzorov tak, že súčet vzdialeností každého vzoru od priemeru je minimálny [5]. Pod pojmom tvar geometrického vzoru bežne rozumieme tie geometrické atribúty, ktoré sa nemenia pri posunutí,



Obrázek 3.3: Pyramídová reprezentácia obrazu.

rotácii a zmene veľkosti telesa [7]. Vzor v k-rozmernom priestore, ktorý pozostáva z n význačných bodov môžeme reprezentovať ako maticu $X : k \times n$. Význačné body sú miesta, ktoré sa v populácii určitých objektov vyskytujú na približne rovnakom mieste a môžeme ich v nich ľahko identifikovať. Napríklad vrcholy špicatých objektov, miesta, kde sa v križovatkách tvaru T stretávajú všetky cesty a tak ďalej.

Definujme, že dva vzory X: $n \times k$ a X': $n \times k$ majú rovnaký tvar, ak je možné previesť jeden na druhú pomocou podobnostnej transformácie

$$X' = \beta X\Gamma + 1_N\gamma^T, \tag{3.16}$$

kde $\beta > 0$ je skalár reprezentujúci zmenšenie alebo zväčšenie, Γ: $K \times K$, $\mid \Gamma \mid = 1$ označuje rotáciu , 1_N je vektor čísel jedna veľkosti N a γ: $K \times 1$ značí posun [7]. Postupne eliminujeme všetky tri komponenty transformácie.

Vzory zarovnávame tak, že minimalizujeme $D = \sum \mid \vec{x_i} - \bar{x} \mid^2$, kde $\vec{x_i}$ predstavuje vzor z množiny a $\bar{x}$ priemerný vzor. Iteratívny algoritmus vhodný pre implementáciu podľa [5] pozostáva z nasledujúcich bodov:

1. Posuň každý vzor tak, že jeho ťažisko je v istom počiatočnom bode.
2. Vyber jeden zo vzorov ako počiatočný priemerný vzor a normalizuj ho, $\mid \bar{x} \mid = 1$.
3. Označ prvý priemerný vzor ako $\bar{x_0}$, aby bolo možné porovnanie v ďalšej iterácii.
4. Zarovnaj všetky vzory so súčasným odhadom priemerného vzoru.

5. Prepočítaj priemerný vzor zo zarovnaných.
6. Zarovnaj vypočítaný priemerný vzor na $\bar{x_0}$ a uprav veľkosť na $\mid \bar{x} \mid = 1$.
7. Ak nie je splnená podmienka ukončenia konvergencie (malá zmena oproti $\bar{x_{i-1}}$), pokračuj bodom 4.

Existujú aj iné prístupy, ktoré môžu produkovať iné rozloženie zarovnaných vzorov [5]. Patrí sem prístup, kedy je povolené meniť naraz zmenu veľkosti aj rotáciu pri minimalizácii D a transformácia každého vzoru do tangenciálneho priestoru.

3.5 Principle Component Analysis

Základným princípom analýzy hlavných komponent (ďalej PCA) je redukovať dimenzionalitu sady dát, ktorá sa skladá z veľkého počtu premenných vo vzájomnom vzťahu tak, aby sa v čo najväčšej možnej miere zachovala variabilita dát v sade. To dosiahneme transformáciou do novej sady premenných, hlavných komponent, ktoré sú nekorelované a sú zoradené tak, že niekoľko prvých zachytáva väčšinu variability obsiahnutej vo všetkých pôvodných premenných [13]. Princíp sa často využíva ku kompresii, ktorá je ale stratová. S PCA súvisí pojem rozptyl. Udáva mieru rozprestrenia dát v dátovej sade. Vypočíta sa podobne ako štandardná odchýlka,

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}, \tag{3.17}$$

kde $\bar{X}$ je priemerná hodnota čísel vo vektore a n je počet čísel vo vektore [19]. Rozptyl štatisticky vyhodnocuje jednorozmerné dáta. V praxi ale často potrebujeme analyzovať prípady, kedy máme k dispozícii viac dimenzii a chceme zistiť, či medzi nimi je nejaký vzťah. Riešením je kovariancia daná vztahom 3.18.

$$cov(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \tag{3.18}$$

Meria sa vždy medzi dvoma dimenziami. Kovariancia jednej dimenzie samej so sebou dáva rozptyl. Pre trojrozmerné dáta (x, y, z) vyhodnotíme $cov(x, y)$, $cov(x, z)$ a $cov(y, z)$. Vypočítaná hodnota nám udáva, ako spolu dáta medzi rozmermi vzájomne súvisia. Ak je hodnota pozitívna, znamená to, že hodnoty v oboch dimenziách stúpajú súčasne. Negatívna kovariancia značí, že ak hodnota v jednej dimenzii rastie, tak v druhej klesá. Nula značí, že dimenzie sú na sebe nezávislé. Pri viacrozmerných dátach z kovariancii jednotlivých dimenzii zostavíme kovariančnú maticu

$$C^{n\times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j)), \tag{3.19}$$

kde Dim_x je x-tý rozmer [19]. Diagonála matice, kedy $i = j$, značí rozptyl dimenzie samej so sebou a hodnoty $cov(a, b)$ a $cov(a, b)$ sú podľa vzťahu 3.18 rovnaké, teda matica je symetrická podľa diagonály.

Pre správne pochopenie fungovania PCA potrebujeme zaviesť pojmy vlastný vektor (*eigenvector*) a vlastná hodnota (*eigenvalue*). Uvažujme štvorcovú maticu A a vektor x. A berieme ako transformačnú maticu, ktorá po násobení s x zmení smer vektora. Existujú určité výnimočné vektory x pre maticu A, ktoré majú rovnaký smer ako Ax a nazývajú sa

vlastné vektory [21]. Ak má matica vlastný vektor, je určite štvorcová. Matica $n \times n$ má práve n vlastných vektorov [19]. Platí rovnosť $Ax = \lambda x$, kde λ sa nazýva vlastná hodnota. Určuje, či je vektor x roztiahnutý, zmenšený, obrátený alebo nezmenený po násobení transformačnou maticou A [21]. Vlastné vektory sú na seba kolmé.

Samotná PCA metóda pozostáva z niekoľkých krokov. Ako prvé musíme od každej hodnoty v dimenzii odčítať priemernú hodnotu dimenzie. Ďalej spočítame kovariančnú maticu podľa (3.19). Pre túto maticu následne nájdeme vlastné vektory a vlastné hodnoty. Vektory normalizujeme tak, aby ich veľkosť bola 1. Jeden z nich prechádza bodmi n-rozmerného priestoru a udáva vzájomný vzťah medzi bodmi dimenzií pozdĺž tohoto vektora [19]. Vektor s najväčšou vlastnou hodnotou predstavuje hlavnú komponentu dátovej sady. Ak zoradíme vlastné vektory podľa veľkosti im prislúchajúcej vlastnej hodnoty od najväčšej po najmenšiu, získame poradie komponent podľa ich významnosti. Pre redukciu dimenzii môžeme vynechať menej významné komponenty, čím dochádza ku stratovej kompresii. Ponechané vektory použijeme ako stĺpce matice, čím vytvoríme vektor rysov (*feature vector*). Výslednú dátovú sadu získame vynásobením transponovanej pôvodnej matice dát (upravenej odčítaním priemernej hodnoty) transponovaným vektorom rysov, podľa vzťahu [19]

$$FinalData = FeatureVector^T \times AdjustedData^T. \tag{3.20}$$

Spätnú projekciu do pôvodných dát vykonáme podľa vzťahu

$$OriginalData^T = ((FeatureVector^T)^{-1} \times FinalData) + OriginalMean, \tag{3.21}$$

kde inverzia vektora rysov *feature vector* pri zachovaní všetkých vlastných vektorov znamená transpozíciu a *OriginalMean* je vektor priemerných hodnôt jednotlivých dimenzii, ktorý sme na začiatku výpočtu odčítali od hodnôt v sade.

3.6 Active Shape Models

V oblasti počítačového videnia požadujeme rozpoznať a interpretovať štruktúry, ktoré vidíme. K tomu potrebujeme modely, ktoré by entity reprezentovali. Takýmto štatistickým modelom sú modely aktívnych tvarov (Active Shape Models). Princíp definovali Cootes a Taylor vo svojej práci Statistical Models of Appearance for Computer Vision [5]. Pracujú s predpokladom, že určité štruktúry v obrazovom zázname sa v priebehu času prirodzene deformujú a my potrebujeme poznať možné variácie týchto zmien. Štruktúry však môžu byť komplexné a obrazový signál môže byť poškodený šumom. Automatizovaný systém aplikuje znalosti získané z predchádzajúcich pozorovaní a obmedzí možné variácie na tie, ktoré sú podľa modelu vhodné a prípustné.

Pre zostrojenie modelu aktívnych tvarov potrebujeme dostatočne veľkú množinu trénovacích vzorov. Skladajú sa z takzvaných význačných bodov, ktoré by mali byť dobre identifikovateľné vo všetkých možných inštanciách sledovanej štruktúry. Každý tvar pozostáva z n význačných bodov v ľubovoľnej dimenzii (najčastejšie 2D a 3D) a reprezentuje jednu variantu možnej deformácie. Je dané, že tvar je invariantný voči podobnostnej transformácii (zahŕňa posun, rotáciu a zmenu veľkosti) [5]. Cieľom práce s modelmi je nájsť taký model, pomocou ktorého môžeme analyzovať nové tvary alebo vytvoriť tvary podobné tým v trénovacej množine.

Uvažujme dvojrozmerný obraz a tvar reprezentovaný n význačnými bodmi ako vektor

$$x = (x_1, x_2, ..., x_n, y_1, y_2, ..., y_n)^T \tag{3.22}$$

s $2n$ položkami. Trénovacia množina bude pozostávať z s takýchto vektorov. Vyjadríme ju ako maticu s rozmermi $2n \times s$ Pred tým, ako sadu štatisticky vyhodnotíme, zarovnáme na seba všetky tvary pomocou Procrustesovej analýzy popísanej v podkapitole 3.4. Ďalej je vhodné redukovať celkovú dimenzionalitu. Využíva sa analýza hlavných komponent rozobratá v podkapitole 3.5. Metóda nájde ku kovariančnej matici jej vlastné vektory a vlastné hodnoty. Vlastné vektory zoradíme zostupne podľa im prislúchajúcich vlastných hodnôt λ_i. Tým získame možnosť aproximovať originálne dáta pomocou modelu s menšou dimenzionalitou t, ktorá zodpovedá počtu t najväčších vlastných vektorov. Počet zistíme tak, že vezmeme súčet všetkých vlastných hodnôt vypočítaných z kovariančnej matice $V_T = \sum \lambda_i$. Ten značí celkový rozptyl dát trenovacej sady. t najväčších vlastných hodnôt vyberieme podľa vzťahu

$$\sum_{i=1}^{t} \lambda_i \geq f_v V_T, \tag{3.23}$$

kde f_v definuje podiel celkového rozptylu, ktorý chceme použiť. Teraz môžeme každý vzor z trénovacej množiny vyjadriť ako [5]

$$x \approx \bar{x} + \Phi b, \tag{3.24}$$

kde $\bar{x}$ označuje priemerný tvar, Φ je matica vlastných vektorov $\Phi = (\phi_1 \mid \phi_2 \mid ... \mid \phi_t)$ a b je t–rozmerný vektor daný vzťahom [5]

$$b = \Phi^T (x - \bar{x}). \tag{3.25}$$

Vektor b definuje parametre modelu a zmenou jeho hodnôt meníme tvar výsledného objektu. Rozptyl i-tého parametru b_i v rámci trénovacej sady určuje hodnota λ_i [5]. Druhá odmocnina λ teda vyjadruje štandardnú odchýlku. Predpokladáme, že pravdepodobnostné rozloženie hodnôt b_i je normálne (Gaussovo). Potom podľa pravidla tri sigma môžeme obmedziť vzťahom 3.26 parameter b tak, že pomocou neho budeme generovať tvary, ktoré pokladáme za prípustné. Pravidlo hovorí o tom, že pri normálnom pravdepodobnostnom rozložení leží 99,7% percent pozorovaných hodnôt leží vo vzdialenosti 3*sigma v oboch smeroch od priemeru. Pre 2*sigma je to 95% a 1*sigma už len 68%.

$$b_i \leq 3\sqrt{\lambda_i}. \tag{3.26}$$

Tvar modelovaný pomocou vektora b premietneme späť od obrazu pomocou podobnostnej transformácie definovanej ako posun X_t, Y_t, rotácie θ a zmeny veľkosti s vzťahom [5]

$$T_{X_t, Y_t, s, \theta}(\bar{x} + \Phi b). \tag{3.27}$$

Kapitola 4

Návrh

Keďže pracujeme s vizuálnymi dátami, je nevyhnutné vytvoriť aplikáciu s grafickým užívateľským rozhraním, ktorá funguje v okne. Po jeho zobrazení užívateľ načíta do programu echokardiografický záznam a vyznačí v ňom časť srdca, ktorú chce sledovať. Medzi jednotlivými snímkami bude metóda optického toku vyhodnocovať pravdepodobný výskyt vyznačených bodov. Kvôli pomerne veľkému šumu a jasovým rozdielom medzi snímkami záznamu nie je metóda vždy úspešná a časom sa môžu sledované body stratiť, či nahradiť im podobnými v inej časti obrazu, ako tej, ktorú považujeme za vyhovujúcu. Preto na kontrolu odhadu využijeme štatistický model – Active Shape Model.

Jeho použitie vyžaduje trénovaciu sadu vopred zaznamenaných vzorov reprezentujúcich sledované štruktúry srdca. Prostredie bude poskytovať nástroje pre načítanie videosúboru a jeho následné prehrávanie. Užívateľ bude v snímkoch na rozličných častiach videa vyznačovať body tvaru a výsledné štruktúry bude ukladať do pamäti programu. Tým získava tvary častí srdca v odlišnej fáze cyklu. Anotovanie týchto štruktúr najodbornejšie a najpresnejšie vykoná študovaný odborník v oblasti rádiológie. Doporučené je naklikať body na hraniciach, kde sa najvýraznejšie mení jas pixelov. Tvary musia byť čo najrozmanitejšie, aby sme maximalizovali počet rôznych prípustných variánt tvarov a deformácií. Zároveň by mal byť počet trénovacích vzorov dostatočný na to, aby postihol rozličné možnosti deformácie. Dôležité pre matematickú analýzu je, aby každý vyznačený vzor mal rovnaký počet bodov, pretože body zapíšeme do matice, ktorá musí mať v každom stĺpci, respektíve riadku, rovnaký počet prvkov. Aby sme užívateľa nenútili dávať pozor na vzdialenosť bodov, vytvoríme krivku prechádzajúcu všetkými anotovanými bodmi a rovnomerne ju rozdelíme. Zaistíme tým rovnaký počet bodov pre všetky prvky trénovacej množiny. Natrénované dáta budeme môcť uložiť do súboru na disk a neskôr ich nahrať a doplniť o nové položky, prípadne získať súbor od iného užívateľa aplikácie.

Vznikne sada vzorov reprezentujúca rozličné prípustné tvary. Pomocou Procrustovej analýzy všetky vzory na seba zarovnáme. Použijeme k tomu analytické riešenie Singular Value Decomposition. Dostaneme tak parametre rotácie a translácie pre najvýhodnejšie zarovnanie. Výsledné body zapíšeme do matice, kde stĺpce budú jednotlivé vzory, $\vec{x} = (x_1, x_2, ..., x_n, y_1, y_2, ..., y_n)^T$. Analýzou hlavných komponent získame vlastné vektory kovariančnej matice dát. Z nich zostavíme vektor rysov a pomocou neho určíme parametre modelu aktívnych tvarov, vektor b. Pre každé b_i určíme prípustný rozsah naučený z trénovacej množine. Sledovaním vyznačených bodov pomocou metódy optického toku generujeme pre každý snímok pravdepodobný nový tvar štruktúry. Určíme preň parametre vektora b a zistíme, či každá hodnota spadá do povoleného intervalu. Ak nie, nájdeme najbližšiu povolenú hodnotu (začiatok alebo koniec intervalu pre b_i) a nahradíme ňou pôvodnú, čím

tvar vrátime naspäť do priestoru odhadnutého za pomoci trénovacej množiny.

Upravené parametre premietneme cez model a podobnostnú transformáciu naspäť do obrazu, čím získame tvar, ktorý je vzhľadom k trénovacej množine považovaný za prípustný. vykreslíme ho ako krivku do obrazu. Toto vykonáme pre každé dva po sebe nasledujúce snímky videa. Aby výpočet prebehol iba jedenkrát, uložíme do pamäti vypočítané tvary korešpondujúce s každým snímkom a umožníme tak zobrazenie vo videu smerom dopredu aj vzad bez viditeľného spomalenia a výpočtov v reálnom čase.

Kapitola 5

Riešenie a implementácia

Aplikácia je napísaná v programovacom jazyku Java SE 7.0. Túto technológiu som zvolil preto, že program vytvorený pomocou nej je prenosný medzi operačnými systémami a zariadeniami. Nájdeme ju nainštalovanú na väčšine stolných a prenosných osobných počítačov. Vývojová sada JDK ponúka v základnej inštalácii knižnicu *Swing* pre pohodlné a prehľadné vytváranie grafického užívateľského rozhrania. Výhodou tiež je, že vzhľad výslednej aplikácie bude rovnaký na všetkých podporovnaných platformách, čo bolo mojím cieľom. Jazyk neustále vylepšuje komunita okolo spoločnosti Oracle. Existujú novšie verzie s drobnými odlišnosťami a pokročilejšími syntaktickými konštrukciami, ako napríklad výrazy lambda kalkulu. Pre moje riešenie nie sú potrebné. Verziu 7 som vybral pre spätnú kompatibilitu zariadení, na ktorých užívatelia neinštalujú najnovšie aktualizácie do svojho systému alebo v zariadeniach nie je možné často meniť softwarovú výbavu.

5.1 Knižnice funkcií

Základná inštalácia prostredia jazyka Java neposkytuje všetky súčasti potrebné k naprogramovaniu môjho riešenia problému. Do implementácie som pridal dve knižnice tretích strán, ktoré nie sú súčasťou štandardnej distribúcie. Potrebné súbory sú dodávané spolu s aplikáciou v adresári *res*.

Prvou z nich je multimediálna knižnica *OpenCV 2.4.10*. Poskytuje metódy pre výpočetne náročné matematické operácie. Aplikácia pomocou nej počíta súčiny matíc, načítava a spracováva snímky videa, zabezpečuje analýzu hlavných komponent a odhad optického toku. Pôvodne vyvinutá pre jazyky C++ a Python sa v roku 2013 dostala pomocou obaľovačov pôvodných funkcií aj do prostredia Java. API je generované automaticky a úzko kopíruje syntax a názvoslovie pôvodných metód [11]. Zariadenie, na ktorom bude program spustený, musí mať okrem dodávaného súboru *opencv2410.jar* stiahnutú a nainštalovanú aj knižnicu funkcií a pri spúšťaní musí parametrom programu java *-Djava.library.path=*path** špecifikovať, kde sa nachádzajú skompilované súbory. Podrobnejší návod k inštalácii a zadaniu parametrov popisuje súbor *Readme.txt* v koreňovom adresári aplikácie.

Druhou knižnicou je *Google JSONSimple*. Pomocou nej aplikácia zostaví textový reťazec v dátovom formáte JSON. Ten je čitateľný ako pre stroj, tak aj pre človeka a pozostáva zo zoznamu bodov uložených ako kľúč-hodnota. Dáta v tomto formáte sú prenosné medzi platformami a technológiami, nie sú závislé na aplikácií ani programovacom jazyku. Funkcie knižnice sú jednoduché na použitie a ich implementácia je optimalizovaná, aby bola dobre škálovateľná aj pre veľké dátové súbory. Moje riešenie pracuje s pomerne malými dáto-

vými sadami, informácie teda ukladá rýchlo a efektívne. Zabezpečuje serializovanie tvarov anotovaných užívateľmi do súboru. Uložené tvary môže kedykoľvek nahrať a ďalej s nimi pracovať, prípadne ich zlúčiť s ďalšími nahratými dátami. Aby užívateľ nemusel pri každom spustení aplikácie generovať ten istý pravdepodobnostný model, existuje JSON súbor s jeho parametrami uložený v adresári aplikácie, a ten sa načíta vždy pri štarte.

5.2 Kód obsluhy programu

Zdrojový kód programu je rozdelený na funkčné balíčky. Združujú súbory definícií tried podľa ich funkcie a použitia. Balíček *App* obsahuje súbor pomenovaný názvom aplikácie – *STEchocard.java* s funkciou *main*. Jej jedinou úlohou je načítať knižnicu OpenCV a spustiť aplikáciu.

Súbor *Constants.java* definuje všetky konfiguračné nemenné hodnoty pre aplikáciu. Z tohto miesta môžeme prispôsobiť vzhľad prvkov grafického rozhrania, nastaviť počet bodov reprezentujúcich výsledný anotovaný tvar, či hladkosť splajn krivky sledovaného tvaru, parameter sigma pre násobenie štandardnej odchýlky tvarov trénovacej množiny, veľkosť okna a výšku pyramídy pre *OpenCV* sledovanie optického toku, cestu k súboru serializovaného modelu a niektoré iné numerické konštanty. Všetky premenné sú doplnené vysvetľujúcim komentárom, ich funkciu však môžeme zistiť priamo z identifikátoru.

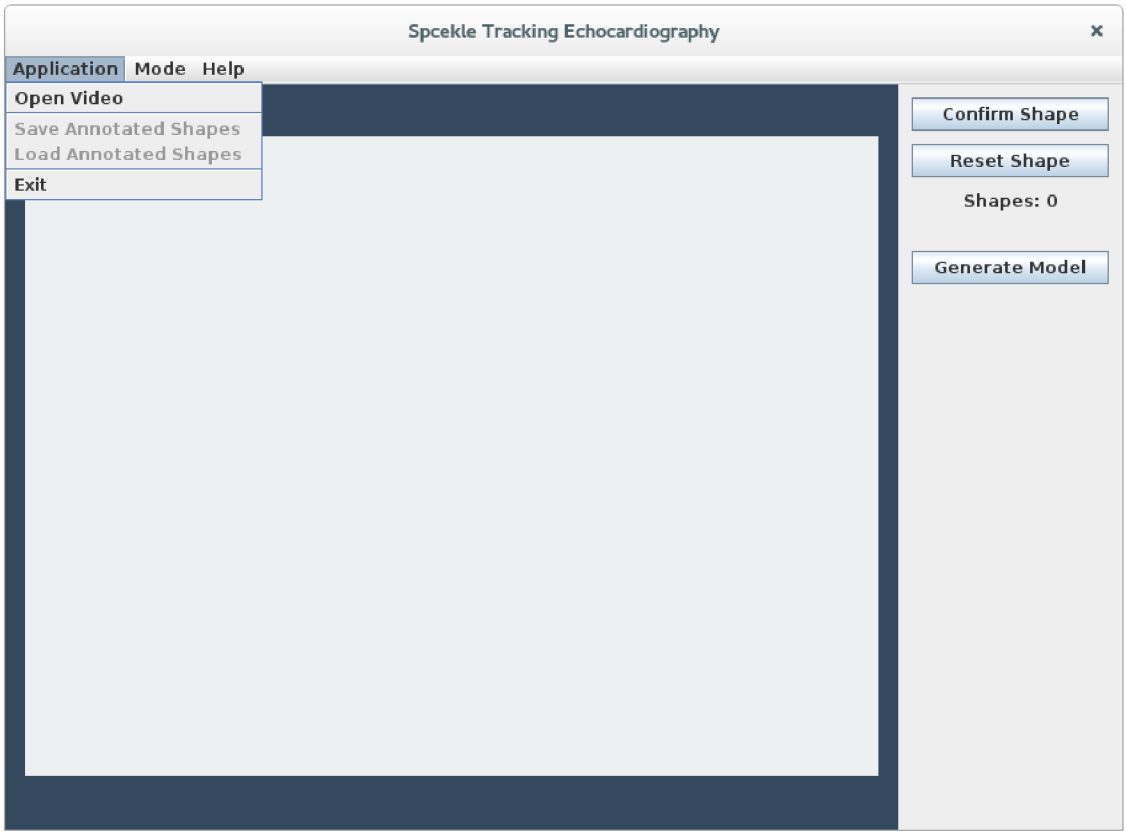
Pri každom spustení programu je vhodné zapamätať si stav, v akom sa pred vypnutím nachádzal. Nenútime tak užívateľa zakaždým opakovať rovnakú postupnosť úkonov, aby mohol začať pracovať. Pre sledovanie pohybujúcich sa a deformujúcich sa tvarov musíme podľa zadania použiť pravdepodobnostný model tvarov trénovacej množiny. S veľkou pravdepodobnosťou užívateľ model natrénuje a vytvorí jedenkrát a bude s ním chcieť pracovať dlhšiu dobu, než ho znovu upraví a doplní. Preto bolo vhodné ihneď po jeho vytvorení uložiť reprezentáciu modelu do súboru. Pre užívateľa môže zostať na prvý pohľad skrytý. Žiadnym spôsobom s ním cielene nemanipuluje, nemusí sa starať o jeho existenciu. Názov a umiestnenie definujeme v rámci kódu konštantou *MODEL_FILE_PATH* v triede *Constants.java* (prednastavená hodnota je adresár *res* v koreni aplikácie). Predpokladá sa, že pri úplne prvom spustení nebude existovať. V tom prípade aplikácia vyzve užívateľa, aby model vygeneroval. Inak načíta obsah súboru vo formáte JSON a nastaví položky objektu triedy *PCA*. Ďalej je už možné tieto dáta používať ku kontrole správnosti odhadnutého tvaru. Znovuvygenerovanie modelu súbor prepíše novými údajmi. Ukladanie aj načítanie využíva funkcie a objekty tried knižnice *JSON Simple*, celé metódy nájdeme v súbore *ModelLoader.java*.

5.3 Prvky užívateľského rozhrania

Program som vytvoril ako grafickú aplikáciu bežiacu v jednom okne. Pozostáva z vrchnej lišty menu, plochy pre video a tlačidiel ovládania programu. Rozloženie prvkov znázorňuje zachytený snímok obrazovky na Obrázku 5.1.

Balíček GUI obsahuje definície tried grafického užívateľského rozhrania. Nájdeme tu najrozsiahlejší súbor – implementáciu hlavného okna a ovládacích prvkov – (*AppInterface.java*). Zaisťuje riadenie celej aplikácie pomocou obsluhy udalostí prvkov rozhrania po interakcii s užívateľom. Ako hlavné okno je viditeľné neustále počas celého behu projektu.

Pomocou štruktúry hlavného okna popíšeme činnosť aplikácie. Dôležitú obslužnú časť programu tvoria položky menu. V sekcii *Application* používame položku *Open Video* pre



Obrázek 5.1: Hlavné okno aplikácie.

načítanie echokardiografického záznamu. Ďalej môžeme nahrať už naanotované tvary *Load Annotated Shapes* zo súboru alebo uložiť novovytvorené *Save Annotated Shapes* na disk pre ďalšie použitie. Sekcia *Mode* umožňuje meniť aktuálnu funkcionalitu aplikácie. Existujú dva režimy pre prácu: anotačný *Annotate Shapes* a sledovací *Speckle Tracking*. V závislosti na nastavenom móde sa mení okrem správania programu aj ovládací panel. Anotačný režim, ako názov napovedá, slúži k vyznačeniu tvarov pre trénovaciu množinu. Pomocou nastavenia pozície vo videu zobrazíme snímky, kde považujeme tvar za významný, vyznačíme ho a potvrdíme stlačením tlačidla *Confirm Shape*. Ak urobíme pri označovaní chybu, neuložené body zmažeme tlačidlom *Reset* a začneme odznovu. Koľko tvarov sme doteraz vyznačili zobrazuje ukazovateľ *Shapes:*. Pre získanie čo najrozmanitejších foriem sledovanej ľavej srdcovej komory a vytvorenie kvalitného modelu musíme bezpodmienečne použiť viac snímkov, najlepšie z viacerých záznamov. Aplikácia preto umožňuje otvárať ďalšie videosúbory bez straty pokroku vyznačovania. Po dosiahnutí požadovaného počtu anotovaných tvarov máme možnosť anotovanú sadu uložiť cez spomenutú položku v menu. Ďalší krok je vygenerovanie modelu. Stlačením tlačidla *Generate Model* prebehne výpočet, parametre modelu sa uložia do interného súboru aplikácie pre načítanie pri ďalšom spustení a prostredie sa prepne do sledovacieho režimu *Speckle Tracking*. Stále máme možnosť vrátiť sa do režimu anotovania cez sekciu *Mode* v menu. Na Obrázku 5.2 vidíme ovládacie prvky sledovacieho režimu. Rovnako ako mód vyznačovania tvarov trénovacej množiny obsahuje prvky prehrávača, ktoré si priblížime v ďalšom texte. Ovládací panel umožňuje potvrdiť, že tvar, ktorý sme na snímku vyznačili chceme ďalej sledovať. Ak sa nám prvotná množina bodov nepodarila podľa našich predstáv, tlačidlom *Reset* ich môžeme vymazať a vyznačiť

ju nanovo. Tlačidlo *Set Tracking* vygeneruje odhady sledovaného tvaru pre každý snímok videa. Od tejto chvíle môžeme pomocou prehrávača sledovať vyznačenú časť srdca meniacu sa v čase. Proces znovu opakujeme pre ďalšie videá, používa sa pre ne rovnaký model.

Ďalší text podrobne rozoberá fungovanie jednotlivých častí popísaného procesu práce s aplikáciou.

Najväčšiu časť hlavného okna zaberá objekt triedy *Canvas.java*. Rozširuje *JPanel* knižnice *Swing*, čo je štandardná plocha pre umiestnenie prvkov GUI. V mojom riešení slúži ako plátno, kam aplikácia vykresluje snímky videozáznamu. Má výšku a šírku snímkov načítaného videa. Po každom nastavení nového snímku sa aktuálne zobrazený obsah prekreslí a ďalej sa zo snímkom už nemanipuluje. Zmena na plátne nastáva pri anotovaní tvarov. Keďže potrebujeme v zázname vyznačiť body tvaru časti srdca, musí plocha reagovať na kliknutie tlačidiel myši. V triede som implementoval obsluhu na udalosť kliknutia ľavým tlačidlom myši na objekt plátna. Súradnice bodu, kam užívateľ klikol, si plátno uloží do vlastnej pamäti naklikaných bodov. Bod ako objekt – inštancia triedy *Point* v sebe zaznamenáva súradnice dvojrozmerného priestoru, x a y. Reakciou na kliknutie do snímku videa je uloženie súradníc do lokálnej pamäti plátna a na ich pozícii sa do snímku videa vykreslí značka v tvare krížika. Toto správanie funguje iba v prípade, že užívateľ už načítal video. Značky sa prekreslia cez snímku. Obsah dočasnej pamäti naklikaných značiek použijeme pri uložení anotovaného tvaru alebo ako definíciu počiatočných bodov určených na sledovanie tvaru štruktúry srdca (popísané v ďalšom texte). Plátno uchováva okrem anotovaných bodov v oddelenej pamäti ešte druhú sadu bodov, a to body splajn krivky. Ak chceme vykresliť kontúry vyznačeného tvaru čo najreálnejšie, nemôžeme prepojiť jednotlivé body priamkou. V mojom riešení ich preto prekladám parametrickou krivkou Catmull-Rom splajn. Množina bodov tohto kubického splajnu vytvorí hladký oblý obrys. Medzi každé dva body umiestňujem priamku, ktorá ich spája. Pri veľkom množstve bodov, ktoré sú blízko pri sebe, sa tvar javí ako oblý a nevidno rovné hrany tvorené úsečkami medzi bodmi. Splajn krivka sa vykresluje iba v móde sledovania tvaru popísaného v ďalšom texte. Do spomenutej druhej sady bodov neukladáme celú splajn krivku, ale iba vopred definovaný počet bodov rovnomerne rozložený po celej jej dĺžke.

K plynulému prehrávaniu videa som rozšíril klasický prvok *JButton* o vlastnosť zapamätania si stavu. Tlačidlo *Play* si po každom kliknutí zmení kontext, v ktorom sa nachádza, a to buď *Play*, alebo *Pause*. Na tlačidle zobrazí príslušný text zodpovedajúci aktuálnemu stavu. Aktivovanie tlačidla v stave Pause spustí nové vlákno, ktoré potom asynchrónne posiela plátnu správy s novým snímkom a žiadosťou o jeho vykreslenie.

Posledným priamo zobrazeným elementom je okno nápovedy. Užívateľovi prezentuje manuál k ovládaniu aplikácie a popis jej funkcií. Je dostupné po kliknutí na položku menu *Help->Manual*.

5.4 Práca s videom

Ako prvý krok pri práci s aplikáciou musí užívateľ nahrať video s echozáznamom srdca. Metódy pre dekompozíciu videosúboru na jednotlivé snímky sú implementované v súbore *VidLoader.java* balíčku Video. *OpenCV* ponúka rozhranie načítavania videa do štruktúry matice. Prístroj snímajúci echozáznamy, ktoré som mal k dispozícii, vytvára video zakódované do formátu H264. Program bol navrhnutý a otestovaný pre tento formát, pri iných kódovaniach nemusí načítavanie videa fungovať správne. Pamäťové obmedzenie virtuálneho stroja Java prostredia musí byť zväčšené, ak pracujeme s dlhšími videosúbormi. Aplikácia bola otestovaná pre dvadsaťsekundové záznamy, stačilo pre ne štandardné nastavenie veľ-

kosti dostupnej pamäti. Získané snímky si zachovávajú svoju pôvodnú veľkosť a farebnú hĺbku. Aby mohli byť vykreslené na zobrazovaciu plochu, musia byť konvertované na bajtový obrázok – objekt triedy *Image*. Dôvodom je, že grafický kontext triedy *JPanel*, mnou rozšírenej na triedu *Canvas*, vykresľuje iba spomenutú reprezentáciu obrazu. Okrem farebnej verzie musíme do pamäti uložiť ešte variantu snímkov v odtieni šedej. Dve po sebe nasledujúce šedotónové matice hodnôt obrazových dát potrebuje pre správny odhad optického toku metóda *OpenCV calcOpticalFlowPyrLK*. Prevod farebných módov zaisťuje funkcia *cvtColor*, rovnako z knižnice *OpenCV*. Uchovávanie nekomprimovaných obrazových dát vyžaduje veľké množstvo operačnej pamäti. Správa pamäti prostredia Java, takzvaný *Garbage Collector*, si s uvoľnením nepoužívaných prostriedkov poradí veľmi dobre a po načítaní nového videa nedochádza k zahlteniu operačnej pamäti. Zoznam starých snímkov sa pred získaním nových zmaže, čím zaniknú odkazy z programu na tieto položky v pamäti.

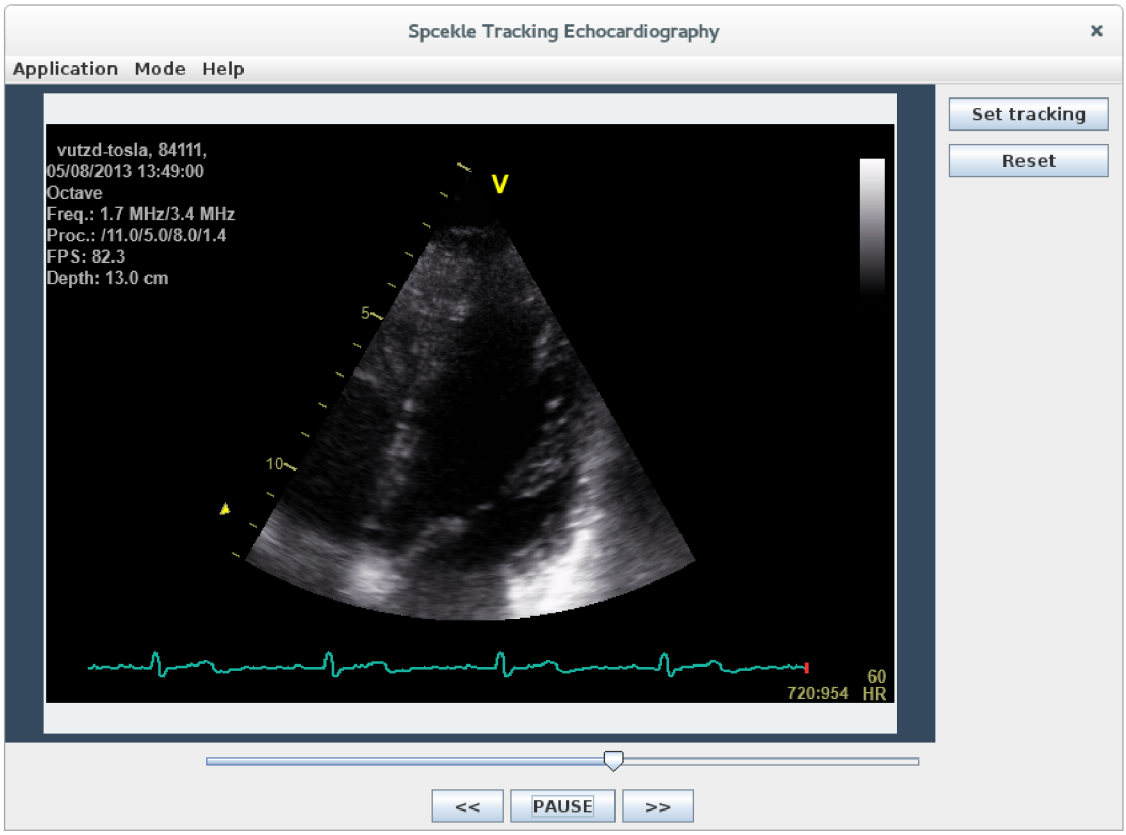
Obe sady snímkov videozáznamu sú uchované v dátovej štruktúre *VidData*. Tá okrem samotných obrazových dát ukladá aj metadáta videa – šírku, výšku a počet snímkov za sekundu. Pre účely prehrávania záznamu obsahuje inštancia objektu *VidData* počítadlo s indexom naposledy prehratého snímku. Používa sa pri spustení prehrávača zo stavu *Pause*, krokovaní alebo nastavení pozície vo videu pomocou posuvníka. Je prístupné pre každý objekt, ktorý pracuje s obrazovými maticami. Trieda je implementovaná podľa návrhového vzoru *singleton*. Po inicializácii existuje v programe vždy práve jedna inštancia, získame ju metódou *getInstance*. Zamedzíme tým existencií viacerých súčasných načítaných videí. Staráme sa iba o jednu načítanú sadu snímkov, a tú pred načítaním ďalšieho videa musíme odstrániť.

Po vykonaní predchádzajúcich krokov si môžeme začať prezerať zvolený echokardiografický záznam. Prostredie prehrávača prezentuje Obrázok 5.2.

Pomocou tlačidiel s označením « a » krokujeme záznam skokovo cez väčší počte snímkov naraz (nastavuje sa pomocou atribútu *FRAME_ADJ_STEP* v *Constants.java*, predvolená hodnota je 5). Posuvník, ako inštancia triedy *JSlider*, mení index aktuálneho snímku v objekte s videodátami a nastavuje príslušný snímok pre vykreslenie na plátno. Tlačidlo *Play/Pause* spustí automatické prehrávanie videa. Programovo sa jedná o nové vlákno, ktorého kód je definovaný v zdrojovom súbore *VidPlayer.java*. Beží oddelene od vlákna grafického prostredia. Volá metódu vykreslenia snímku na plátno a aktualizuje pozíciu ukazovateľa na posuvníku. Takéto správanie by bolo na každom výkonnom počítači nevyhovujúce. Vykresľovanie snímkov by trvalo veľmi krátku dobu a celé video by sa prehralo veľmi rýchlo. Mojím riešením je uspať vlákno na krátku dobu, čím oneskorím vykreslenie každého snímku o danú konštantnú časovú dobu. Tá závisí na snímkovacej frekvencii (framerate) videa a má hodnotu $(1/framerate) * FRAMERATE\_ADJUSTMENT$. Prednastavená hodnota *FRAMERATE_ADJUSTMENT* v súbore konštánt je 400 a bola získaná čisto experimentálne. Na rozličných počítačoch bude nutné prispôsobiť rýchlosť prehrávania zmenou tejto hodnoty. Vlákno prehrávača beži pokiaľ má k dispozícii snímky alebo pokiaľ nebolo prerušené stlačením tlačidla *Pause* a nastavením príznaku *playing* sledovaného v cykle na hodnotu *false*. Ak užívateľ nastavil sledovanie tvaru, prehrávač spolu s novým snímkom pripraví plátnu vysledovaný tvar pre zobrazenie.

5.5 Spracovanie tvarov

Hlavným pilierom aplikácie je manipulácia s množinami bodov. Aj keď existujú automatizované systémy detekcie štruktúr, hlavne pre medicínske použitie, trénovaciu sadu vytváram

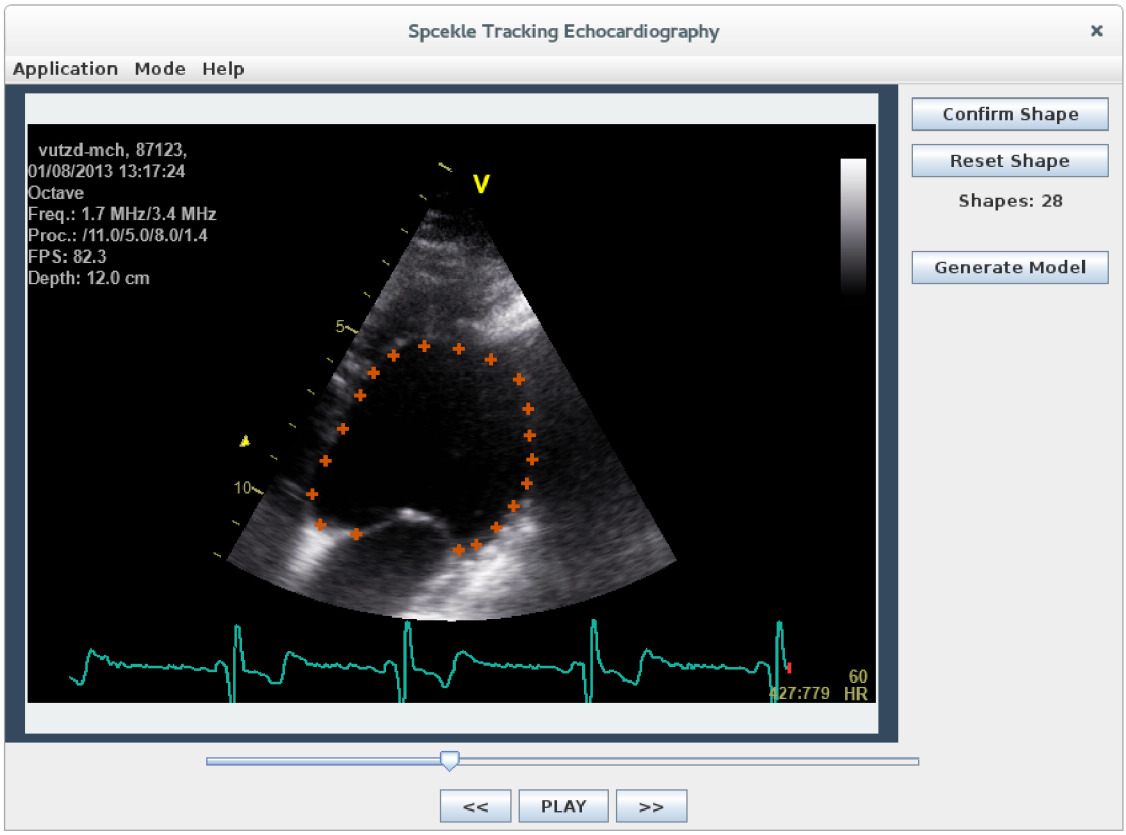


Obrázek 5.2: Prehrávanie videa.

ručne. Jedna sada 2D súradníc tvorí jeden tvar. Užívateľ ho nadefinuje stlačeniami ľavého tlačidla myši na plátno so snímkom videa alebo vznikne výpočtom. Všetky musia byť uložené do príslušných štruktúr a môžu mať viacero reprezentácií (matica pre PCA, zoznam *ArrayList* bodov *Point*, matice pre optical flow, dvojstĺpcová matica, kde riadky predstavujú body). Nasledujúce odstavce priblížia postup spracovania dát a ich prezentáciu do aplikácie.

Balíček *ShapeProcessing* obsahuje sedem tried. Základnú implementáciu reprezentácie tvaru nájdeme v definícii triedy *Shape*. Uchováva dve množiny bodov. Užívateľom anotované body a body krivky kopírujúcej vyznačený tvar rovnomerne rozložené po celej jej dĺžke. Prvú množinu získame klikaním ľavým tlačidlom myši na zobrazovaciu plochu prehrávača s videom. Predstavuje surové vstupné dáta pred spracovaním. Príklad vidíme na Obrázku 5.3.

Jeden tvar môžeme vyznačiť pomocou ľubovoľného počtu bodov. Musia ale nasledovať za sebou presne v poradí, v akom bol tvar definovaný. Nie je možné označiť jednu pozíciu, hneď po nej druhú a v ďalšom kroku sa vrátiť naspäť a doplniť ďalšiu medzi ne. Nefungovalo by totiž správne preloženie bodov interpolačnou krivkou. V takom prípade dôjde k prekríženiu hrán krivky, a to je nežiadúci jav. Program nezoraďuje body podľa niektorej zo súradnicových osí. Riadi sa poradím, v ktorom boli vložené súradnice do zoznamu. Môžeme si to predstaviť ako kreslenie tvaru jedným ťahom štetca. Ďalším limitujúcim faktorom je smer anotovania. Všetky tvary musia vzniknúť pridávaním bodov buď v smere, alebo proti smeru hodinových ručičiek. Užívateľ musí túto konvenciu dodržať aj pri pre-



Obrázek 5.3: Anotovanie tvaru do trénovacej množiny.

pínaní režimov programu, smer postupnosti bodov anotovaných tvarov musí zodpovedať smeru tvaru definovaného pre sledovanie pohybu. Zvlášť je treba dať pozor na to, aký smer postupnosti majú body tvarov uložené v súbore a na aký smer je natrénovaný aktuálne používaný a uložený model. Zaveďme jednotnú konvenciu *v smere* hodinových ručičiek, ak sa explicitne neurčí inak.

Vyznačené body sú väčšinou navzájom rôzne vzdialené. Jeden tvar tak dokážeme vyjadriť pomocou rôzneho počtu bodov. Pri matematickej analýze tvarov, popísanej v ďalšom texte, však musí byť počet súradníc rovnaký pre každý z nich. Požiadavku je dôležité splniť, aby sme mohli prvky trénovacej množiny porovnávať. V mojej práci som použil riešenie, kde som anotované body jedného tvaru interpoloval parametrickou kubickou krivkou Catmull-Rom. Interpolačná vlastnosť zaisťuje, že prechádza všetkými zadanými pozíciami. Ak riadiace body anotované úžívateľom nie sú navzájom od seba príliš vzdialené a nie je ich malý počet, tak krivka reprezentuje pomerne presne tvar, ktorý žiadal uložiť. Zároveň musíme vziať do úvahy fakt, že krivka v diskrétnom priestore pixelov, kde pozície udávajú celé čísla, nemusí vždy byť úplne oblá. Pri meraní dĺžky krivky alebo vykresľovaní spájame úsečkami dva po sebe nasledujúce body. Ak je počet bodov medzi dvomi riadiacimi pozíciami malý, budeme vidieť jednotlivé úsečky. Hladká krivka pozostáva z množstva dostatočne krátkych úsečiek, ktoré nedokonalosť ľudského oka nepostrehne a vníma celok ako oblý. To súčasne zodpovedá sledovanej štruktúre srdca, ktorá je tiež oblá a neobsahuje viditeľné lomené hrany kontúr na videozázname. Počet bodov Catmull-Rom splajnu medzi dvomi riadiacimi bodmi som empirickým testovaním nastavil na 20 (je možné zmeniť v *Constants.SPLINE_SAMPLES_PER_SPAN*). Krivka je vo videu s daným rozlíše-

ním dostatočne hladká a oblá. Každá geometrická krivka v 2D priestore má svoju dĺžku v určitých jednotkách. Nie je preto problém rozdeliť ju na rovnako veľké úseky určeného počtu. Ak zadáme rozdelenie na 2 úseky, spočítame celkovú dĺžku a vydelíme ju číslom 2. Máme vypočítanú veľkosť oboch úsekov. Týmto spôsobom môžeme každý tvar reprezentovať tromi bodmi – počiatočným (prvým anotovaným), bodom vo vzdialenosti jedného úseku na krivke a koncovým (posledným anotovaným). Získané body uložíme napríklad do matice a vykonáme porovnanie s inými tvarmi pomocou niektorej matematickej metódy. Obecne tak dostaneme *počet intervalov*, na ktoré tvar delíme, *plus jeden* bodov. Tri sú však pre vyjadrenie očakávaného tvaru málo, preto v mojom riešení delím interpolačnú krivku na väčší počet rovnakých intervalov a každý tvar definujem o jedno väčším počtom bodov. Hodnotu som zvolil po sérii pokusov. Pre veľkosť videa 636x434 a priestor, ktorý sledovaná srdcová komora zaberá, nie je vyhovujúce použiť viac ako 30 bodov. Zistenie vhodného počtu je uvedené v kapitole 6. Preložená krivka celkom presne vyjadruje požadovanú štruktúru. Metódy pre výpočet bodov krivky a rozdelenie na rovnako dlhé intervaly obsahuje statická trieda *CatmullRom*. Implementovaný program rovnomerne rozložené body uloží do inštancie triedy *Shape*, atribútu *SplinePoints*.

Potvrdením množiny anotovaných bodov zaháji metóda *serializeShape* inštancie triedy *Shapes* výpočet interpolačného splajnu, rozdelí ho rovnomerne na zadaný počet bodov a objekt Shape pridá do svojho zoznamu. Rozširuje Java kolekciu *ArrayList*. Implementuje metódy ukladania a nahrávania tvarov trénovacej množiny modelu. Pred uložením dát na disk vezme sadu tvarov a transformuje ju na znakový reťazec formátu JSON. Ten je čitateľný ako pre stroj, tak aj pre človeka a pozostáva zo zoznamu bodov uložených ako kľúč-hodnota. Do súboru ukladáme iba nespracované body. Ak v budúcnosti zmeníme počet splajn bodov reprezentácie tvaru alebo získame trénovaciu množinu v súbore od užívateľa s inak nastaveným parametrom, musíme byť schopní vytvoriť nový model. Po pridaní nových tvarov do načítanej trénovacej množiny, ktorá rozdeľovala krivky na iný počet intervalov, ako máme súčasne nastavený, vytvorenie modelu nebude možné. Takto zakaždým interpolujeme anotované body odznovu. Zoznam pre uloženie alebo načítanie bodov zo súboru vytvárame pomocou objektov knižnice *JSONSimple* – *JSONArray*, *JSONObject* a ich metód. Výsledný objekt typu pole zapíšeme metódou *writeJSONString* ako textový reťazec do súboru. Jeho umiestnenie a meno si užívateľ vhodne zvolí pomocou dialógového okna, nie sú pevne stanovené. Opačný postup funguje pri načítaní serializovaných tvarov zo súboru. Ten znovu cez dialógové okno zvolí užívateľ sám. Objekt *JSONParser* svojou metódou *parse*, ktorá ako jediný parameter akceptuje textový reťazec v JSON formáte načítaný z disku, rozloží serializovaný tvar na ďalšie objekty. Týmto postupom získame naspäť všetky údaje (súradnice bodov) a môžeme zostaviť reprezentáciu tvaru v prostredí Java. Užívateľ tak má možnosť anotovať tvary v jednom sedení a svoj postup si uložiť. Pri ďalšom sedení si nahrá uloženú množinu bodov a pridá k nej nové, čím rozširuje databázu tvarov. Dosiahneme teda väčšiu variabilitu a môžeme použiť súbory s tvarmi od iných užívateľov pracujúcich s našou aplikáciou.

Ďalej prechádzame k samotnému spracovaniu tvarov a vytvoreniu modelu. Prvý krok je zarovnať na seba anotované tvary tak, aby suma druhých mocnín Euklidovských vzdialeností bola minimálna. Vhodnou technikou je *Procrustova analýza* popísaná v sekcií 3.4. Súbor *Procrustes.java* definuje triedu statických metód. Jedna z nich je hlavná, *analyze*, a používa všetky ostatné. Ako parameter jej predáme sadu tvarov trénovacej množiny, objekt *Shapes*. Od tejto chvíle ich budeme reprezentovať maticami $N \times 2$, kde riadky predstavujú body a stĺpce súradnice x a y. Použijeme ich pre zjednodušenie a zefektívnenie násobenia matíc funkciami knižnice *OpenCV*. Transformáciu na matice vykoná jedna z metód

sady pomocných funkcií definovaných v triede *MatUtils*. Zarovnanie musí prebiehať vzhľadom k jednému referenčnému reprezentantovi trénovacej množiny. Na začiatku procesu nie je dôležité, ktorý z tvarov to bude, preto som zvolil prvý tvar z danej množiny. Ďalej ho budeme nazývať priemerný. V ďalších iteráciách sa vyhodnotí nový priemerný tvar, ktorý nahradí inicializačný. Proces prebieha v cykle. Na rozdiel od iteračného riešenia v [5] počítam parametre transformácie analyticky pomocou metódy *Singular Value Decomposition*. Určí parametre podobnostnej transformácie, ktorá garantuje najmenšiu možnú sumu Euklidovských vzdialeností korešpondujúcich bodov, teda najlepšie možné zarovnanie jedného tvaru B na priemerný tvar A. Vypočítajú sa geometrické ťažiská tvarov ako priemerná hodnota súradníc bodov, a to oddelene pre smer x a y. Ťažiská odpočítame od súradníc príslušného tvaru, čím množiny bodov posunieme do počiatku súradnicovej sústavy. Nasleduje krok výpočtu kovariančnej matice H priemerného a vyšetrovaného tvaru. Vyjadrí vzájomnú závislosť súradníc tvarov pre osy x a y. H veľkosti 2×2 rozložíme pomocou SVD na dve matice reprezentujúce hlavné poloosy (vektory) a maticu koeficientov (skaláry) určujúcu zmenu veľkosti poloosí priemerného tvaru, ako popisuje sekcia 3.1. Z výsledku dekompozície

$$SVD(H) = U\Sigma V \tag{5.1}$$

podľa [8], ktorý vychádza z článku *A Method for Registration of 3-D Shapes* autorov Besla a McKaya, vyrátame rotačnú maticu R ako

$$R = VU^T. \tag{5.2}$$

Tá má znovu rozmer 2×2. Keď ňou vynásobíme každý bod, vyjde nám tvar rotovaný o uhol, kedy sú skúmané množiny bodov k sebe navzájom najbližšie, pričom tvar a veľkosť zostáva zachovaná. Ešte potrebujeme určiť translačnú zložku podobnostnej transformácie. Zo vzťahu

$$t = -R * tazisko\_priemerneho\_tvaru + tazisko\_zarovnavaneho\_tvaru \tag{5.3}$$

vzíde vektor posunu. Výsledný výpočet transformácie dostaneme ako:

$$priemerny\_tvar = R * zarovnavany\_tvar + t \tag{5.4}$$

Proces opakujeme pre každý tvar trénovacej množiny. Ďalej sa už postup zhoduje s navrhnutým riešením Cootesa a Taylora [5]. Znovu vyrátame priemerný tvar zarovnania a porovnáme ho s predchádzajúcim pomocou sumy chýb – druhých mocnín Euklidovských vzdialeností vzájomne korešpondujúcich bodov. Ukončovacou podmienkou zarovnávacieho cyklu je, že sa pozície priemerných tvarov v dvoch po sebe nasledujúcich iteráciách nezmenia.

5.6 Analýza tvarov

Teraz, keď máme spracovanú trénovaciu množinu, prejdeme k analýze tvarov. Doposiaľ boli vyznačené štruktúry reprezentované maticami každá zvlášť. Pre výpočet PCA potrebujeme zložiť všetky dokopy a vytvoriť maticu s rozmermi $n \times m$, kde n je počet tvarov a m počet bodov tvaru krát počet dimenzií bodu (2D). Vychádzame z reprezentácie navrhnutej Cootesom a Taylorom [5], kedy každý tvar definujeme vektorom hodnôt x nasledovaných hodnotami y: $[x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]$. Získali sme určitú reprezentáciu všetkých možných tvarov. Jej veľkou nevýhodou je príliš veľká dimenzionalita, ktorá by urobila výpočet

náročným a neefektívnym. Preto sa snažíme počet dimenzií redukovať, lepšie povedané minimalizovať. K tomu nám pomôže metóda *PCAComputeVar* knižnice *OpenCV*. Ako parametre jej zadáme vytvorenú maticu tvarov, prázdnu maticu, kam uloží vypočítaný priemerný tvar v trénovacej množine, maticu pre uloženie hodnôt vlastných vektorov (eigenvectors) a množstvo variability, ktoré chceme z vypočítaných hodnôt zachovať. Od veľkosti posledného parametra závisí, akú časť celkového rozptylu (variance) pozorovaného v trénovacej množine modelom pokryjeme. Vypočítané vlastné vektory sú zoradené zostupne podľa veľkosti prislúchajúcej vlastnej hodnoty. Každá vlastná hodnota vyjadruje mieru variability obsiahnutej pozdĺž vlastného vektora. Celková suma všetkých vlastných hodnôt zodpovedá celkovej variabilite pozorovanej v trénovacej množine. Vektory, ktoré sú na konci zoznamu, majú pravdepodobne malé vlastné hodnoty, teda reprezentujú malú variabilitu. Môžeme ich pokladať za šum a výsledný model by ovplyvnili iba v malej miere. V mojom riešení som ako posledný parameter funkcie zvolil hodnotu 0.98, čo reprezentuje 98% rozptylu v rámci matice natrénovaných tvarov. Zvyšné 2% nebudú brané v úvahu a nepoužijú sa ani vlastné vektory, ktoré im prislúchajú. Tento princíp je podstatou kompresie pomocou PCA v niektorých algoritmoch spracovania multimediálnych dát. S použitím modelu už nebude možné zreštaurovať každý anotovaný tvar úplne presne, ale iba ako jeho aproximáciu, čo pre moje riešenie nepredstavuje žiadne výrazné obmedzenie. Verzia *OpenCV* pre platformu Java na rozdiel od C++ alebo Python implementácie nevráti spolu s vlastnými vektormi ich vlastné hodnoty. Tie som musel počítať samostatne metódou *eigen* triedy *Core*. Hlavným parametrom, z ktorého určí výsledné hodnoty, je kovariančná matica. Vyrátal som ju pomocou funkcie *calcCovarMatrix*. Výsledné vlastné hodnoty však boli príliš veľké a nedali sa použiť v kombinácií s vlastnými vektormi. Z [3] som zistil, že musím použiť príznak *Core.COVAR_SCALE* v kombinácií s ostatnými príznakmi funkcie pre výpočet kovariancie, aby boli hodnoty upravené pre získané vlastné vektory. S nastaveným príznakom funkcia vydelí každý prvok kovariančnej matice počtom vzorkov, ako je uvedené v [3]. Konečne tak volaním *eigen* získame požadované vlastné hodnoty. Ich počet však môže byť väčší ako počet vektorov. Prebytočné vlastné hodnoty preto nepoužijeme. Pre model nám zostáva definovať povolený rozsah parametrov. Po projekcií ľubovoľného tvaru do priestoru daného vlastnými vektormi získame objekt reprezentovaný vektorom b s hodnotami parametrov modelu. Veľkosť tohto vektora zodpovedá počtu vlastných vektorov a hodnôt. Každý parameter môžeme podľa pravidla tri sigma limitovať iba na určité povolené hodnoty. Tým garantujeme, že tvar, ktorý hodnotami reprezentujeme, zodpovedá niektorému tvaru trénovacej množiny. Parametre $b_1, \ldots, b_n$ obmedzíme tak, že ich absolútna hodnota bude menšia ako $N * \sigma_i$, teda $N * \sqrt{eigenvalue_i}$. Sigma predstavuje štandardnú odchýlku, N je násobiteľ udávajúci obmedzenie modelu, bude vysvetlené v ďalšom texte. Ako príklad pre použitie pravidla 2 sigma, kedy zabezpečíme 95% celkovej variability sady, zmeníme konštantu *SIGMA_MULTIPLIER* v *Constants.java*. K modelu pridáme dva vektory horných a spodných hraničných hodnôt parametrov b a model máme hotový. Pripomeňme, že vlastné vektory, hodnoty, priemerný tvar a hraničné hodnoty sa zapíšu do súboru modelu a budú použité pri ďalšom spustení programu.

Keď máme model vygenerovaný alebo načítaný zo súboru, môžeme prejsť do fázy sledovania. Stlačením tlačidla *Generate Model* sa prostredie automaticky prepne do sledovacieho režimu, inak použijeme menu *Mode*. Pred sebou máme snímok videa. Snažíme sa nájsť taký, kde je okraj tvaru, ktorý chceme v obraze sledovať, celistvý a výrazný. Metóda počítania optického toku tak na začiatku dokáže vypočítať novú polohu sledovaného bodu jednoduchšie a presnejšie ako v obraze, kde sa príliš významne menia hodnoty jasu, predovšetkým vplyvom šumu. Body tvaru anotujeme rovnako ako v režime trénovania modelu. Smer

vyznačovania zachovávame rovnaký ako ten pre trénovaciu množinu, teda v smere hodinových ručičiek. Riadiace body aplikácia znovu preloží interpolačnou krivkou a rovnomerným rozdelením vytvorí reprezentáciu tvaru s odpovedajúcim počtom bodov. Po dokončení anotovania tlačidlom *Set Tracking* spustíme metódu triedy *OpticalFlow calcOpticalFlow*. Ako parametre jej predáme anotovaný tvar, štruktúru s šedotónovými snímkami videa a zoznam, kam uloží vypočítané body. Začne prechádzať snímky videa postupne v poradí, ako nasledujú za sebou. Ak snímok, ktorý sme určili ako počiatočný anotovaním tvaru, nie je na pozícii nula, po vypočítaní všetkých tvarov na snímkach v smere dopredu sa vráti na počiatočnú pozíciu a spracuje snímky od počiatočného miesta dozadu. Na smere, v ktorom video spracovávame, nezáleží, dôležité je, aby snímky nasledovali jeden po druhom, keďže počítame zmenu medzi dvoma snímkami. Oba tvary, vstupný aj výstupný, majú formu matice *OpenCV MatOfPoint2D*. Jedná sa o dvojkanálovú reprezentáciu $1 \times n$, kde n je počet bodov a súradnice x,y ležia v dvoch odlišných kanáloch. V cykle voláme metódu *calcOpticalFlowPyrLK* knižnice *OpenCV* a dostávame odhadnutý tvar. Sledovanie bodov optického toku pomocou funkcie knižnice môže mať definovanú vlastnú veľkosť okna. Zvolíme ju parametrom funkcie odhadu toku vždy s nepárne číslo. Tu môžeme nastaviť aj výšku pyramídy obrazu. Popis výberu vhodných hodnôt približuje kapitola 6. Vysledovaný tvar nemôžeme vykreslit. Najskôr však musíme skontrolovať, či metóda optického toku našla správne body. Kvôli zašumenému obrazu môže vzniknúť tvar, ktorý sa nepribližuje žiadnemu v trénovacej množine. Taký prehlásime za nevyhovujúci. Do obrazu videa však musíme umiestniť body odhadnutej štruktúry a pokračovať vo vyhodnocovaní odhadov na ďalších snímkoch. Riešením je aproximovať získaný tvar prvkom z trénovacej množiny, ktorý sa mu najviac podobá. Jednoduchý a efektívny iteračný algoritmus navrhujú Cootes a Taylor v [5]. Prispôsobením na môj problém som navrhol nasledujúci algoritmus:

1. Nastav najbližší tvar na priemerný tvar trénovacej množiny
2. Zarovnaj posunom a rotáciou tvar vypočítaný pomocou metódy optického toku na najbližší tvar
3. Premietni zarovnaný tvar do priestoru modelu
4. Skontroluj hodnoty parametrov tvaru po premietnutí do priestoru modelu a orež prípadné presahy mimo povolené intervaly
5. Premietni tvar naspäť do priestoru súradníc 2D priestoru
6. Spočítaj sumu Euklidovských vzdialeností bodov tvaru od korešpondujúcich bodov aktuálne najbližšieho tvaru
7. Ak je suma menšia ako v predchádzajúcej iterácii, nastav tvar ako najbližší tvar a vráť sa na bod 2
8. Tvar nastav na najbližší tvar

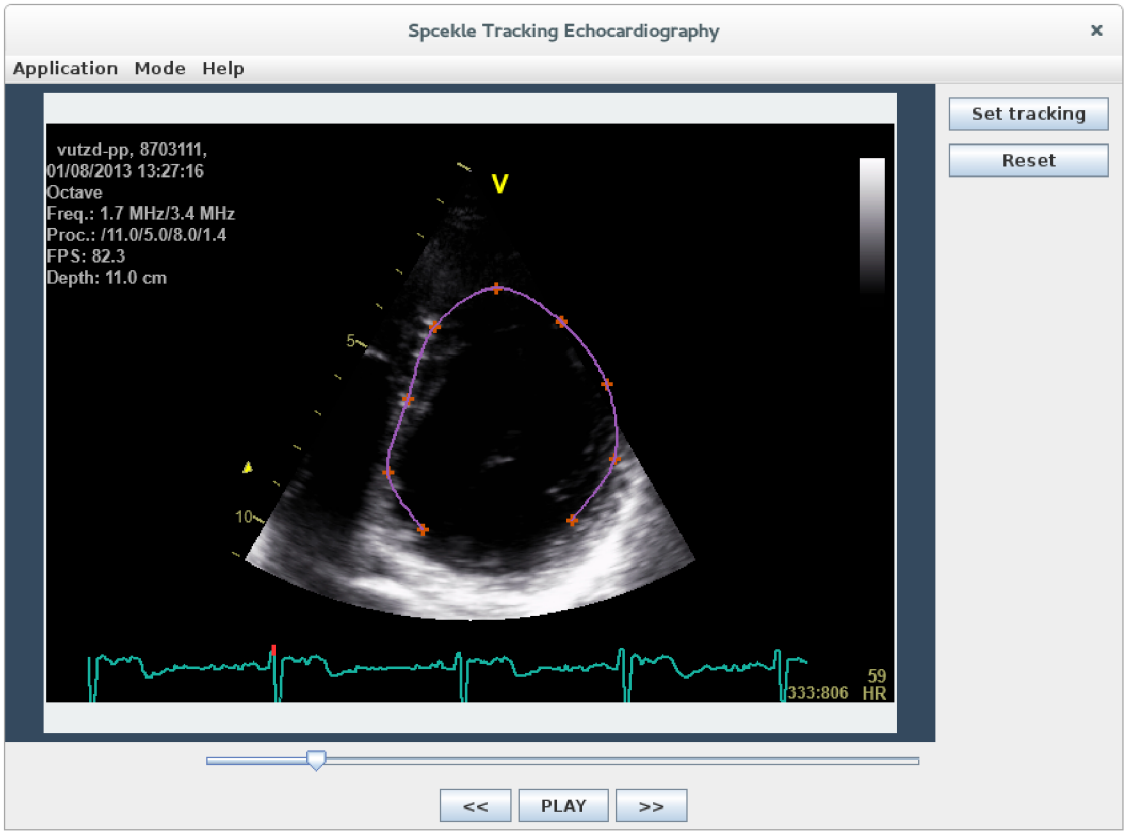
Po dokončení algoritmu máme k dospozícii množinu bodov, ktorú pokladáme za reprezentáciu prípustného tvaru srdcovej komory pozorovanej v rámci trénovania modelu. Odhadnutý tvar najskôr zarovnáme na priemerný tvar trénovacej množiny. Premietneme ho do priestoru modelu použitím funkcie *PCAProject* knižnice *OpenCV*, čím získame vektor hodnôt parametrov modelu $[b_1, b_2, \ldots, b_n]^T$. V predchádzajúcej časti sme si vysvetlili, že pri vytvorení modelu z trénovacej množiny vzniknú dolné a horné hranice hodnôt pre

každý parameter b_i. Skontrolujeme, či premietnutý objekt spadá do spomenutých intervalov. Ak niektorá hodnota prekračuje zadané medze, nahradíme ju príslušnou hraničnou variantou. Spätnú projekciu do maticového vyjadrenia súradníc x,y dokončíme duálnou funkciou k projekčnej, a to *PCABackProject*. Výsledok uložíme ako aktuálne najlepší tvar. V cykle týmto spôsobom porovnávame pôvodný odhadnutý tvar s aktuálne najlepším, ten postupne aktualizujeme. Zjemňuje a spresňuje sa výber reprezentanta trénovacej množiny najlepšie vyhovujúceho odhadnutému tvaru. Pokračujeme dovtedy, pokiaľ sa suma vzdialeností zmenšuje oproti poslednej iterácii. Ak body vrátené z metódy počítania optického toku *OpenCV* reprezentujú prípustný tvar, algoritmus prebehne iba dvakrát, inak konverguje v jednotkách iterácií.

Proces sledovania pokračuje nasledujúcim snímkom tak, že na vstup metódy optického toku vložíme výsledok popísaného algoritmu. Ďalší odhad bodov vychádza práve z neho, čím znížime riziko šírenia chybne odhadnutých bodov mimo tvary trénovacej množiny a usmerníme sledovanie požadovaným smerom. Počiatočný tvar, ktorého pozíciu a deformáciu chceme v zázname sledovať, môžeme vyznačiť v ľubovoľnom snímku videa. Najskôr sa spracujú snímky od tohoto indexu do konca videa, potom sa program vráti na počiatočný index a vyhodnotí tvary naspäť smerom k začiatku videa.

Výsledný tvar uložíme do zoznamu všetkých odhadov, ktorý má rovnaký počet položiek ako je počet snímkov videa. Ako sa zobrazí sledovaný tvar na echokardiozázname môžeme pozorovať na Obrázku 5.4. Ku každému snímku prislúcha práve jeden vysledovaný tvar. Prehrávanie a vykresľovanie v režime *Speckle Tracking* je tak rýchle, sledovanie tvarov medzi snímkami sa vypočíta iba raz a zostávajú uložené v pamäti programu pokiaľ nestlačíme tlačidlo *Reset* alebo užívateľ neprepne aplikáciu do módu *Annotate Shapes*. Umožní nám to používať posuvník prehrávača videa. Posuny pomocou neho sú veľmi rýchle, aj o niekoľko desiatok snímkov a výpočet tvarov v reálnom čase by nemusel stíhať dodávať body tvaru pre vykreslenie. Rovnako tak môžeme krokovať záznam skokovo cez viac snímkov bez toho, aby sme nárazovo museli vypočítavať všetky tvary medzi dvomi pozíciami bez toho, aby sa vykreslili.

V predchádzajúcich častiach textu boli spomenuté štyri rozličné formy reprezentácie anotovaného alebo sledovaného tvaru: Java zoznam *ArrayList* bodov *AWT Point*, matica *OpenCV Mat* $m \times n$ hodnôt typu double, kde m riadkov predstavuje n súradníc bodov – x,y, dvojkanálová matica *OpenCV MatOfPoint2D* $m \times 1$ hodnôt typu double, kde kanál 1 nesie hodnotu x a kanál 2 hodnoty y a nakoniec matica *OpenCV Mat* pre analýzu PCA s rozmermi $1 \times 2k$, kde k je počet bodov tvaru a hodnoty sú uložené v poradí $[x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_k]$. Zoznam *AWT Point* sa vykresľujú na plátno ako anotačné značky a splajn krivka, *OpenCV Mat* $m \times n$ matice sa zarovnávajú pomocou Procrustovej analýzy, *OpenCV MatOfPoint2D* $m \times 1$ spracováva počítanie optického toku a vektory $[x_1 i, x_2 i, \ldots, x_k i, y_1 i, y_2 i, \ldots, y_k i]$ sa po zložení do matice všetkých tvarov premietnu do modelu. Vzájomné prevody medzi všetkými spomenutými formami zabezpečujú statické funkcie triedy *MatUtils*. Nealokujú nové štruktúry, výsledky uložia do objektov predaných ako parametre, čo nám umožní znovupoužitie už alokovaných.



Obrázek 5.4: Sledovanie tvaru.

Kapitola 6

Testovanie a výsledky

Overenie funkčnosti riešenia pozostáva z dvoch fáz: vyhodnotenie priemernej chyby vo videozázname a následná intuitívna vizuálna kontrola. Podľa výsledkov merania chyby nastavím parametre programu a skontrolujem, ako fungujú pri bežnom používaní aplikácie užívateľom. Ako prvé si ale predstavíme sadu testovacích videozáznamov. K dispozícii som mal 9 rôznych videí, ktoré mi poskytol vedúci práce. Sú zakódované do formátu H264 a zabalené do kontajnera MPEG-4. Každý z echozáznamov má rozlíšenie 636×434 pixelov a dĺžku 20 sekúnd. Pri zobrazovacej frekvencii 25 snímkov za sekundu to predstavuje viac ako 500 obrazových matíc v jednom videosúbore. Tvary, náklon a poloha sledovanej srdcovej komory sa podľa predpokladu v jednotlivých videách líšia. Veľkosť srdca napríklad ovplyvňuje fyzická veľkosť sledovaného jedinca. Rôzna je tiež kvalita záznamu vzhľadom k množstvu šumu. Niektoré videosúbory zobrazujú kontúry srdcovej štruktúry pomerne nejednoznačne a jas pixelov na predpokladaných okrajoch sa výrazne mení. Jedno z videí je poškodené približne v polovici záznamu, nebolo preto použité k vyhodnoteniu výsledkov, ale iba k získaniu tvarov do trénovacej množiny z nepoškodených snímkov. V každom zázname som vybral 7 snímkov, ktoré sa od seba čo najviac líšili v tvare srdcovej komory. Vyznačil som priemerne 20 anotačných bodov na jeden tvar, aby vypočítaná interpolačná krivka kopírovala štruktúru čo najpresnejšie. Voľbou menu *Application->Save Annotated Shapes* som ich uložil do súboru *training_shapes.json*. Medzi každými dvoma riadiacimi bodmi vzniklo pomocou krivky Catmull-Rom 20 bodov interpolačného splajnu. Rovnomerným rozdelením celkovej dĺžky krivky preloženej vyznačenými riadiacimi pozíciami dostávame sady význačných bodov. Množstvo výsledných reprezentujúcich bodov bude predmetom optimalizácie v ďalšom texte. Celkovo som vytvoril 63 tvarov. Distribúciu v 2D priestore vidíme na Obrázku 6.1. Body rovnakej farby prislúchajú tomu istému tvaru.

Testovanie prebiehalo v sledovacom režime. Aby sme mohli začať testovať úspešnosť navrhnutej sledovacej metódy, musíme nastaviť kľúčové parametre. Sú to veľkosť okna pre výpočet optického toku, výška pyramídy optického toku, množstvo variability z trénovacej množiny a počet bodov reprezentujúcich jeden anotovaný tvar. Cieľom je nastaviť ich tak, aby vzájomná kombinácia minimalizovala vzdialenosť bodov sledovaného tvaru na začiatku a konci videa s ohľadom na fázu srdcového cyklu. Srdcové sťahy sa po určitej perióde opakujú. V každom videu anotujeme na niektorom z prvých snímkov jeden tvar a ku koncu záznamu nájdeme odpovedajúci snímok po vykonaní niekoľkých cyklov srdca.

Na základe rozboru riešenia a dostupných informácií z literatúry som stanovil možné hodnoty sledovaných parametrov, ktorých kombináciu som vyhodnocoval. Redukoval som tým prehľadávaný priestor kandidátnych riešení, ktorý exponenciálne rastie. Veľkosť inte-

Obrázek 6.1: Množina bodov reprezentujúcich tvary trénovacej množiny.

gračného okna pre odhad optického toku doporučuje [2] nastaviť v rozsahu 5×5 až 15×15 pixelov. Aby zostali zachované nepárne hodnoty veľkosti strany integračného okna, použil som zväčšovanie s krokom 2 pixely. Predpokladám použitie väčšieho okna, pretože pohyb pixelov je pomerne rýchly a jas sa v okolí sledovaného bodu často mení. Ďalej ten istý zdroj uvádza, že nedáva zmysel voliť výšku pyramídy väčšiu ako 4. Pre naše video by bolo rozlíšenie snímku na štvrtej úrovni iba 40×28 pixelov, čo je veľmi nízke rozlíšenie pre zaznamenanie pohybu na echokardiografickom zázname. Vybral som rozsah výšky pyramídy od 0 do 3, čo zodpovedá štyrom úrovniam s rozlíšeniami 636×434, 318×217, 159×109 a 80×55 pixelov. Množstvo variability získanej z trénovacej množiny môžeme obmedziť pomocou pravidla 3 sigma. Vynásobením odmocniny i-tej vlastnej hodnoty číslami 3 a -3 získame interval parametra b_i, v ktorom ležia hodnoty formujúce tvary trénovacej množiny. Predpokladáme normálne rozloženie pravdepodobnosti výskytu hodnôt a konštanta 3 zabezpečí zisk 99,97% celkovej variability. Čím menšie je násobiace číslo, tým obmedzenejší model dostaneme, bude sa meniť menej výrazne. Pri testovaní som použil okrem hodnoty 3 aj dvojku, čo zodpovedá 95% pozorovanej celkovej variability v rozsahu parametra b_i. Posledným nastaviteľným parametrom je počet bodov reprezentujúcich tvar. Viac riadiacich bodov nasledovaných za sebou určí intuitívne tvar presnejšie podľa toho, ako ho pri vyznačovaní vidíme. V skutočnosti môže stačiť menej bodov a tvar bude zodpovedať našim predstavám. S nárastom počtu pozícií, ktoré metóda optického toku sleduje, však môže narásť pravdepodobnosť, že niektorý z bodov sa nepodarí nájsť presne a vzdiali sa z celkového tvaru. Vplyv na presnosť odhadu som skúmal nastavením počtu bodov v rozmedzí 5 až 14 s krokom 3.

Pre každé použiteľné video som podľa krivky vyjadrujúcej priebeh srdcového vyklu, ktorá sa nachádza v spodnej časti echozáznamu, zvolil snímok okolo začiatku a konca videa. Oba vyjadrujú približne rovnakú fázu cyklu. Ako počiatočnú pozíciu záznamu som sa snažil vybrať záber, na ktorom je dobre vidieť kontúry štruktúry, ktorú budeme sledovať. Medzi spomenutými dvomi pozíciami prebehlo niekoľko cyklov srdcovej činnosti. Pre každý beh som definoval body ručne. Tým som zaviedol určitú náhodnosť, ktorá existuje pri používaní aplikácie. Metóda optického toku pracuje s pixelmi, je teda takmer nemožné, aby užívateľ zakaždým zvolil rovnaké súradnice. Testujeme tak schopnosť modelu obecne kontrolovať tvary. Samotné vyhodnocovanie presnosti odhadu tvaru prebiehalo tak, že po-

čiatočný anotovaný tvar som porovnal s tým na konci sledovania. Kritériom bola priemerná vzdialenosť na jeden bod. Celková suma Euklidovych vzdialeností korešpondujúcich bodov musí byť ešte vydelená počtom týchto bodov. Poloha komory srdca na zázname sa v priebehu snímania mohla zmeniť. Množinu bodov na konci pozorovania som zarovnal na počiatočnú anotovanú pomocou rotácie a translácie. Výsledná chyba tak nezávisela na polohe a natočení. Cieľom je priemernú chybu minimalizovať.

Na každom z ôsmich videozáznamov som spustil šesť behov. Tri pre jeden počiatočný snímok v ejekčnej fáze srdcového cyklu a tri pre druhý v relaxačnej. Cyklus vyhodnocovania aplikoval postupne všetky možné kombinácie parametrov a spočítal priemernú chybu na jeden bod. To predstavuje $4 * 5 * 2 * 4 = 160$ výpočtov všetkých tvarov medzi zadanými pozíciami v jednom behu. Celkovo tak testovacia procedúra vyskúšala $48 * 160 = 7680$ iterácií odhadu postupnosti tvarov sledovaných na zázname.

Vyhodnotil som, ktorá kombinácia parametrov produkuje najmenšiu chybu. Každá zo 160 konfigurácií dosiahla v každom zo 48 behov určitú odchýlku od požadovaného tvaru. Ovplyvniť ju môže skutočnosť, že tvar a veľkosť srdca sa medzi cyklami nevýrazne mení. Tiež predpokladáme, že odhad korešpondujúcej počiatočnej a koncovej fázy v cykle nie je presný. Zistil som priemernú hodnotu a medián. Zoradil som výsledky zostupne vzhľadom k priemernej chybe. 20 najlepších riešení spolu so štatistikami predstavuje Tabuľka 6.1. Položka *params* je vo formáte *veľkosť_optical_flow_okna, výška_optical_flow_pyramídy, násobok_štandardnej_odchýlky, počet_bodov_tvaru*. Kompletnú tabuľku výsledkov nájdeme v súbore *res/vysledky_komplet.csv*. V rovnakom adresári je uložený aj kód testovacej procedúry, *OpticalFlowTest.java*.

Params	**Priemer [px]**	**Medián [px]**	**Min [px]**	**Max [px]**
9x9/0/2/8	12,92	17,35	6,60	37,94
9x9/1/2/8	13,41	16,24	6,02	42,77
9x9/2/2/8	13,43	16,68	4,94	38,29
9x9/3/2/8	13,885	19,43	8,12	43,57
7x7/0/2/5	14,28	15,35	6,83	36,80
11x11/0/2/8	14,37	14,10	6,27	30,38
11x11/1/2/8	14,39	13,12	5,08	34,87
11x11/2/2/8	14,73	16,11	8,76	34,50
11x11/3/2/8	14,75	16,81	7,74	39,20
13x13/0/2/8	14,77	14,55	6,77	29,29
13x13/1/2/8	14,81	13,88	3,10	34,25
13x13/2/2/8	14,87	12,54	5,69	36,43
13x13/3/2/8	14,95	14,96	5,50	36,36
15x15/0/2/8	14,96	13,72	4,17	31,43
15x15/1/2/8	14,99	14,37	5,01	36,25
15x15/2/2/8	15,00	13,28	4,54	35,98
15x15/3/2/8	15,09	12,78	5,61	31,45
7x7/1/2/5	15,12	19,99	4,78	44,83
7x7/0/2/11	15,26	18,09	7,56	35,87
7x7/1/2/11	15,29	17,67	7,92	37,03

Tabulka 6.1: Najlepšie riešenia vzhľadom k priemernej chybe na jeden bod.

Už na prvý pohľad je zrejmé, že výška pyramídy optického toku má iba malý vplyv na výsledok. Integračné okná rovnakej veľkosti sú združené pri sebe, líšia sa o veľmi malú hodnotu. Fenomén bol pozorovaný aj v kompletnej výsledkovej tabuľke. Z hľadiska efektívnosti výpočtu som zvolil ako výsledný parameter hodnotu 1, čo predstavuje dve úrovne – plné rozlíšenie a jedno podvzorkované.

U veľkosti okna vidíme, že s nárastom veľkosti sa zvyšuje priemerná chyba, ale klesá medián. Zmeny predstavujú asi dva pixely, čo je pomerne zanedbateľná hodnota. Vo zvyšku tabuľky trend pokračuje a so zvyšovaním veľkosti okna sa chyba zväčšuje. Zvolil som hodnotu 13×13, pretože má stále pomerne malú priemernú chybu a najnižší medián.

Podľa očakávania obmedzenie modelu násobkom štandardnej odchýlky sigma nedovolilo produkovať tvary príliš odlišné od priemerného. Takmer všetky hodnoty 3 sa objavili v druhej polovici tabuľky výsledkov. Ak by sme model obmedzili ešte viac, nemuseli by sme postrehnúť všetky zmeny tvaru a javil by sa ako málo sa deformujúci. Kompromisom bolo zvoliť násobenie štandardnej odchýlky od priemeru hodnotou 2 a získať 95% všetkých možných tvarov.

Počet bodov reprezentujúcich tvar výrazne menil výsledné chyby. Hodnota 5 posunula všetky kandidátne riešenia, v ktorých figurovala, až do druhej polovice tabuľky. Pre správny odhad tvaru ich počet nestačil. Naplnila sa pôvodná obava, že s rastúcim počtom bodov narastá aj riziko chybného určenia hľadaného bodu a odchýlenie sa od očakávaného tvaru. Ako vidíme v tabuľke, veľká väčšina najlepších riešení bola dosiahnutá pomocou 8 sledovaných bodov, preto som ju použil vo výslednom nastavení.

Intuitívna vizuálna kontrola prebiehala tak, že na jednom z prvých snímkov som vyznačil obrys ľavej srdcovej komory. Nechal som sledovať tvar do konca videa a následne spustil prehrávnanie. Pozoroval som, či vypočítaná kontúra vyznačená vo videozázname meniaca sa v čase zodpovedá tvaru zaznamenanému na snímku. Program správne reagoval na zmeny polohy, veľkosti aj tvaru. Výsledky pre každé video záviseli na tom, ako sa menil jas pixelov okolo okraja štruktúry. Pre niektoré pokusy sa tvar zdeformoval a program začal sledovať susedné body, ktoré nepatria do vyznačenej štruktúry, ako napríklad chlopeň. Keďže sa po premietnutí do priestoru modelu jednalo o prípustný tvar, aplikácia ho vyhodnotila ako správny a zobrazila ho. Nastali aj prípady, kedy poloha a celkový tvar štruktúry boli vyznačené správne, ale tvar bol väčší ako na začiatku. Výsledné riešenie v závere nedosiahlo presnosť, akú som na začiatku predpokladal.

Kapitola 7

Závěr

Výsledkom mojej práce je grafická aplikácia *STEchocardiography*. Implementuje prehrávač echozáznamov, anotačný editor tvarov srdcovej komory a sledovanie vyznačeného tvaru vo videu. Užívateľ na začiatku procesu definuje čo najrozličnejšie tvary. Môže použiť predpripravenú sadu uloženú v súbore alebo ju získať prenosom od iného užívateľa. Následne vygeneruje model alebo použije už uložený z predchádzajúceho sedenia. V sledovacej fáze na začiatku ručne anotuje štruktúru, ktorú chce nechať sledovať. Pyramídová implementácia metódy optického toku vo variante Lucas – Kanade odhaduje zmenu polohy daných bodov medzi dvoma po sebe nasledujúcimi snímkami. Kvôli výraznému šumu v obraze na zázname aplikácia kontroluje, či nové body vytvárajú štruktúru podobnú niektorej z pozorovaných v trénovacej fáze. Tu zapojíme do riešenia pravdepodobnostný model. Odhadnuté body premietneme do priestoru modelu. Ak parametre nespadajú do intervalu definujúceho tvary trénovacej množiny, orežeme ich na hraničné hodnoty a tvar premietneme naspäť do priestoru obrazu. Vypočítané tvary sa vykresľujú do videa počas prehrávania, posunu alebo krokovania snímkov.

Riešenie som vyhodnotil na sade 8 echozáznamov. Pozostávalo z merania priemernej chyby pixelu medzi tvarom na začiatku záznamu v určitej fáze srdcového cyklu a na konci videa v rovnakej fáze. Hodnoty sa pohybovali okolo 13 pixelov, čo je pomerne veľa vzhľadom k rozlíšeniu videa. Záviseli aj na presnosti počiatočného anotovania. Podľa výsledkov som nastavil vhodné parametre programu. Sledovaním záznamu som pozoroval, ako správne kopíruje vypočítaný tvar ten skutočný. Program korektne sledoval posun štruktúry a deformácie. Zobrazené výsledky sa často ale po pár cykloch odlišovali od skutočnosti zmenou v jednej časti tvaru. Je to spôsobené tým, že aj optickým tokom nesprávne odhadnutý bod tvoril vo výsledku vyhovujúci tvar pozorovaný v trénovacej množine. Najčastejšie sa to stalo v blízkosti pohybujúcej sa chlopne alebo tam, kde sa náhle zmenil jas veľkej množiny bodov.

K lepšiemu výsledku by mohlo prispieť predspracovanie obrazu potlačením šumu, ktorý často prispel k strate presnej polohy sledovaného bodu. Vyhladenie pomocou mediánového filtra zníži rozdiely jasu pixelov, čím priemeruje hodnoty okolia, čo by mohlo viesť k presnejšiemu určeniu polohy bodu. V rámci riešenia by sa ešte dal otestovať odhad pomocou väčšieho integračného okna metódy optického toku. Mohli by sme viac obmedziť variabilitu modelu, ale tým by sa nemuseli postihnúť všetky možné tvary pre všetky echozáznamy.

Ďalším výsledkom je technická správa, ktorá popisuje teoretický rozbor použitých technológií a riešenia spolu s implementačnými detailmi a návodom na použitie. Približuje metódu voľby optimálnych parametrov riešenia a vyhodnotenie výsledkov.

Literatura

[1] Aires, K. R. T.; Santana, A. M.; Medeiros, A. A. D.: *Optical Flow Using Color Information*. ACM New York, 2008, ISBN 90-247-2689-1.

[2] Bouguet, J. Y.: Pyramidal Implementation of the Lucas Kanade Feature Tracker, Description of the algorithm [online]. `http://robots.stanford.edu/cs223b04/algo_tracking.pdf`.

[3] Bradski, G.; Kaehler, A.: *Learning OpenCV*. O'Reilly Media, 2008, ISBN 978-0-596-51613-0.

[4] Béder, I.; et al.: *Fyziológia človeka*. Vydavateľstvo Univerzity Komenského v Bratislave, 2004, ISBN 80-223-2028-5.

[5] Cootes, T. F.; Taylor, C. J.: Statistical Models of Appearance for Computer Vision [online]. `http://www.face-rec.org/algorithms/AAM/app_models.pdf`, 2004.

[6] Fleet, D. J.; Weiss, Y.: *Handbook of Mathematical Models in Computer Vision*. Springer, 2006, ISBN 0-387-26371-3.

[7] Goodall, C.: Procrustes Methods in the Statistical Analysis of Shapes. Journal of the Royal Statistical Society. Series B (Methodological). ročník 53, č. 2, 1991: s. 285–339.

[8] Ho, N. K.: Finding potimal rotation and translation between corresponding 3D points [online]. `http://nghiaho.com/?page_id=671`, 2013-05-10 [cit. 2015-05-19].

[9] Horn, B.: *Robot Vision*. MIT Press, Cambridge, Massachusetts, 1986, ISBN 9780262081597.

[10] House, D. H.: Spline Curves [online]. `http://people.cs.clemson.edu/ dhouse/courses/405/notes/splines.pdf`, 2014-04-14 [cit. 2015-05-17].

[11] Itseez: OpenCV now supports desktop Java [online]. `http://opencv.org/opencv-java-api.html`, 2013-02-15 [cit. 2015-05-17].

[12] Javorka, K.; et al.: *Lekárska fyziológia*. Vydavateľstvo Osveta, Martin, 2001, ISBN 80-8063-023-2.

[13] Jolliffe, I. T.: *Principal Component Analysis, Second Edition*. Springer, 2002, ISBN 0-387-95442-2.

[14] Joy, K. I.: On-Line Geometric Modeling Notes: CATMULL-ROM SPLINES [online]. `http://graphics.cs.ucdavis.edu/ joy/ecs278/notes/Catmull-Rom-Spline.pdf`, 2008-02-19 [cit. 2015-05-17].

[15] Lucas, B. D.; Kanade, T.: An iterative image registration technique with an application to stereo vision [online]. http://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf, 1981.

[16] Mráz, P.; et al.: *Anatómia ľudského tela*. Slovak Academic Press, Bratislava, 2004, ISBN 80-89104-57-6.

[17] Nekula, J.; et al.: *Radiologie*. Univerzita Palackého v Olomouci, 2005, ISBN 80-244-1011-7.

[18] Purves, D.; et al.: *Neuroscience*. Sunderland (MA): Sinauer Associates, druhé vydání, 2001, ISBN 0878937420.

[19] Smith, L. I.: A tutorial on Principal Components Analysis [online]. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002-02-26 [cit. 2015-01-13].

[20] Stoylen, A.: Basic ultrasound, echocardiography and Doppler for clinicians [online]. http://folk.ntnu.no/stoylen/strainrate/Ultrasound, 2014-12 [cit. 2015-01-09].

[21] Strang, G.: Introduction to Linear Algebra, 4th Edition [online]. http://math.mit.edu/ gs/linearalgebra/, 2009-07-01 [cit. 2015-01-13].

[22] Trefethen, L. N.; BauIII, D.: *Numerical linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics, 1997, ISBN 978-0-89871-361-9.

[23] Warren, D. H.; Strelow, E. R.: *Electronic Spatial Sensing for the Blind: Contributions from Perception*. Springer, 1985, ISBN 90-247-2689-1.

[24] Wedel, A.; Cremers, D.: *Stereo Scene Flow for 3D Motion Analysis*. Springer London, 2011, ISBN 978-0-85729-965-9.

[25] Yuksel, C.; Schaefer, S.; Keyser, J.: Parameterization and Applications of Catmull-Rom Curves. *Computer Aided Design*, ročník 43, č. 7, 2011: s. 747–755.